The Uses of Interactive Computer Graphics

For Solving Differential Equations

by

O. P. Brereton

Ph.D.

1978

ORIGINAL COPY IS TIGHTLY BOUND AND TEXT IS CLOSE TO THE EDGE OF THE PAGE

ACKNOWLEDGEMENTS

The work for this thesis has been carried out at the Computer Centre, University of Keele under the supervision of Dr. H. H. Greenwood and Mr. D. J. Battye.

I would especially like to express my gratitude to Dr. Greenwood for his guidance throughout all stages of the project.

I would also like to thank all the Computer Centre staff and, in particular, Dr. P. Collis and Mrs. A. F. Grundy, for their valuable programming advice.

In addition I am grateful to Mrs. C. Goulding for typing the thesis.

CONTENTS

ABSTRACT

The aim of this thesis is to investigate the use of interactive computer graphics in a system of programs designed to solve ordinary and partial differential equations.

The first chapter is an overall introduction to the work carried out and is followed by a chapter containing a general survey of interactive computer graphics, including its development, available hardware and software, properties of interactive graphics systems and a selection of the current areas of application.

The third chapter describes the particular hardware and software used with the ICL 4130 at the University of Keele.

The types of ordinary and partial differential equations selected for inclusion in the set of interactive graphics programs are described in Chapter 4, followed by a discussion of the methods of solution chosen for each type of problem. The methods themselves are given in greater detail in the appendix. The chapter ends with a review of the more important ancilliary activities performed within the program. These are: curve fitting, the solution of linear algebraic equations, visual representation of 3-dimensional surfaces and the real time input of algebraic expressions.

In Chapter 5 the design of an interactive graphical system for solving mathematical problems is considered with particular reference to differential equations. The design principles are implemented in the three main programs, described in Chapter 6, for the solution of ordinary and partial differential equations.

In Chapter 7 some of the advantages of interaction and computer graphics when solving differential equations are illustrated by example problems for each of the three programs.

The final chapter contains a discussion of the conclusions and recommendations, based on the experience gained in the use of the programs written, on the uses of interactive computer graphics for solving differential equations.

1. INTRODUCTION

The main objective of this work is to examine the ways in which interactive computer graphical methods can be used to control and interpret the solution of ordinary and partial differential equations.

The particular types of equations chosen as a basis for the study provide many of the different problems that arise when solving differential equations. These equations fall into three categories:

1. A set of simultaneous first order ordinary differential equations either of the initial-value or boundary-value type. The equations are solved using a numerical method and the solutions are displayed graphically. The parameters describing the problems and solution method as well as those controlling the form of the graphical output can be changed interactively at prescribed points in the computation.

2. Parabolic partial differential equations in one or two space dimensions or elliptic partial differential equations in two dimensions. The methods of solution used are of the finite difference type and for two dimensional problems the region of solution must be rectangular. The solution of a two-dimensional problem in this category is a function of two variables which is displayed visually as a rotatable surface projected onto the plane of the display screen.

3. Two-dimensional elliptic partial differential equations defined over an arbitarily shaped region and solved using a finite element method. In this category the areas in which interactive computer graphics make a major contribution are in the description of the boundary of the region of solution, the shape, size and position of the finite elements and in the presentation of the solution surface.

The central part of the thesis is concerned with the description and development of interactive computer graphics in mathematical problem solving and with the design of a system for solving differential equations. Since it is not intended to engage in differential equations research as such, the three experimental programs presented for the categories listed above incorporate only established methods of solution.

2. INTERACTIVE COMPUTER GRAPHICS

This chapter presents a survey of the development and current 'state of the art' in the broad field of interactive computer graphics. A selection of the graphical displays and input devices presently available are described, followed by a discussion of the properties of interactive computer graphics systems and their most common areas of application.

2.1. Definition and Development

A study of the use of systems of interactive computer graphics presents at the outset, clear problems in terminology. In a field where developments in both hardware and range of applications are rapid, any definition is likely to be, at best, ephemeral. It has been decided for present purposes, therefore, to consider the terms 'computer graphics' and 'interaction' separately, as a clarifying preliminary to the more central concern of this study: interactive computer graphics.

The term 'computer graphics' is used to describe the manipulation and display of pictorial representations of information or objects, using a computer. Manipulation can include graph scaling, object rotation (in two or three dimensions), windowing and many other of the tasks necessary for the control of the display. The display itself may be produced using devices such as a lineprinter, digital plotter or graphical display unit, the latter usually incorporating a cathode ray tube.

'Interaction' is used to describe some form of on-line man/machine communication in which, for example, a scientist, engineer or other computer user is able to communicate with a program, at selected points during its run, via an on-line terminal. Interactive computer graphics combines these two facilities allowing on-line use of a computer via a graphical display terminal.

The cathode ray tube was first used as a high speed computer output device with the machine 'Whirlwind 1'[22] in 1950. However, modern interactive computer graphics systems have evolved from work carried out by the United States Department of Defence in the mid and late 1950's. The SAGE aircraft surveillance system[35], for example, used a display screen to give a visual indication of the positions of aircraft within, and adjacent to, the United States. It functioned as a visual information retrieval system, in which an operator could locate and request expansion of information about any given target, using a light-pen. Development in 1956 and the early 1960's included the first interactive computer graphics system for calculation and design. Known as the 'Sketchpad' project[55], this development by I. E. Sutherland at M.I.T.'s Lincoln Laboratories was, like earlier work, funded by the United States Department of Defence. It involved the use of various ring data structures for topographical as well as geometrical definition of display objects and allowed the initiation and control of running programs using input devices such as the light-pen and function keyboard. 'Sketchpad' is generally considered to be the first major breakthrough in the field of interactive computer graphics.

In the early 1960's the largest computer-aided-design development group using interactive graphics was at General Motors, where the prototype of the IBM 2250 display system was produced. As a result of this work, IBM developed the software necessary to enable this display to be used with their new 360 computer range. Since then there have been widespread developments in the field of interactive computer graphics both in the United States and in Great Britain. Much of this work is described in 'Advanced Computer Graphics'[46], 'Pertinent Concepts in Computer Graphics'[23] and 'Computer Graphics'[47].

2.2. Interactive Graphics Systems: Hardware and Software

In this section the most important types of interactive graphics systems in current use, including the most widely used hardware and software, are described. These systems fall into three main categories: single-user stand-alone systems, conventional time-sharing systems, and satellite graphics systems.

Single-user systems comprise a processor unit, display unit and output equipment, and other peripheral equipment wholly dedicated to one or more specific graphics applications programs. These dedicated systems are expensive specialized machines usually found in research and university establishments or in large industrial organisations.

Time-sharing systems can be either of a conventional terminal sort or of a satellite sort. The conventional terminal systems are considerably cheaper per user than the single-user systems but suffer from slow response times, making dynamic graphics almost impossible. The satellite approach to computer graphics originated in an attempt to design a time-sharing system with a response time similar to that of a single-user graphics system. This was achieved by means of a small general-purpose computer incorporated in the terminal, but involves a large amount of complex software to enable the terminal's local processor to communicate with the central processor.

2.2.1. Hardware

There are two elements of interactive graphics hardware: visual display units and their associated input devices.

A visual display unit commonly consists of a cathode ray tube, although other media (the 'Plasma Panel[7], which stores and displays pictures using small gas discharge cells, for example) are sometimes employed. The cathode ray tube uses electrical fields to generate a finely focussed

high speed beam of electrons, which is deflected to different parts of the screen surface where it generates a visible trace.

Most graphical display units have an alphanumeric keyboard which can be used either to supply data to a program or to issue commands. Alphanumeric keyboards do not however permit direct communication with the display, where it is often useful to be able to point to positions on the screen or to draw in lines or symbols. Devices which make this direct interaction possible include the Stanford Research Institute (SRI) 'mouse', electronic stylus tablets and the light-pen.

The SRI 'mouse'[20,21] consists of a small box containing two potentiometers mounted orthogonally, each having a wheel attached to its shaft. As the box is moved over a scaled surface in front of the display, the wheels turn the shafts to resolve the motion into two orthogonal components, which are transmitted to the screen electronically. The newly selected position is thus transferred to the screen which serves as a scale representation of the surface being moved over.

The term 'tablet' refers to a surface, usually separate from the display, which can be drawn on with a stylus. There are several types of tablets, but the type described here is that developed by the Rand Corporation[17]. It consists of a flat drawing surface ten inches square, having 1024 copper lines parallel to each axis, to form a grid. Each line carries a uniquely coded signal that can be picked up by the stylus to give its position on the grid.

The mouse and the tablet are both positioning devices. In contrast, the light-pen is primarily a pointing device which, when pointed at an item on the screen, generates information from which the item can be identified. However, the light-pen can only be used to position items if the necessary

tracking is performed by software programs. The light-pen associated with the Keele single-user system is described in the next chapter.

To produce a picture on a display screen it is necessary to create a display file: that is, a series of machine instructions, together with relevant data. Programs which use a display file compiler can be written in high level languages such as Algol 60, Fortran or PL/1 although these languages are not entirely adequate for this task. Algol 60 is weak in data manipulation, contains no string-handling facilities and offers only arrays as a means of building data structures. Fortran, in addition to these drawbacks has obscure and out-moded input and output facilities. PL/1 offers more comprehensive data manipulation facilities, including more flexible data structures. However, all three languages have the common failing that they are not designed for conversational use.

Conversational languages such as JOSS[3] and BASIC[51] have very limited program and data structure facilities, making them more suited to simple problems than to large interactive ones. Of the better known conversational languages, APL[34] is the best suited to large scale problem solving as it provides comprehensive number and vector handling facilities, although LISP[60] and CPL[4] are also used for this purpose.

A number of languages have been designed specifically for use in an interactive graphics environment. These include ISLAND[11], which is an interactive graphics language for mathematical computation, and IMAGE[45], a language that places particular emphasis on control structure, picture description and device-independent input and output. W. K. Gilio[27], in a paper comparing Fortran, Algol, PL/1 and APL as bases for structured display programming, introduces a model language for interactive display programming (GRIP). Further examples of interactive graphics languages can be found in the Proceedings of the ACM Symposium on Graphics Languages[16] held in 1976.

Instructions, written in one of these high-level languages are converted to visual displays through the medium of display file compiler programs. These compiler programs receive graphical function calls from user programs, which typically will be a request to the compiler to add a graphical entity, such as a point or line, to the display file. The coordinates of the point or line are passed on as parameters for the call, and are used by the compiler to construct the appropriate sequence of display instructions, which are added to the display file.

It is often necessary to transform (i.e. scale, translate, rotate, window, clip, etc.) pictures, subpictures or individual symbols. The transformation required is applied to the data structure representing the whole picture, before it is displayed on the screen. The file produced before transformations are carried out is not displayed prior to the transformation itself and so is called a 'pseudo'-display file. Once the transformation process has taken place, the 'pseudo'-display file becomes the 'transformed' display file.

A large quantity of data is often necessary to define a picture to be displayed. This is generally stored in the form of a data structure which is scanned by the display file compiler when producing the pseudo-display file. A data-structure is a model or representation of information or objects, and will often contain more detail than can be displayed at any one time. An overview of data-structures and their suitability for computer graphics is provided by R. Williams[62]. He describes languages for creating and manipulating data-structures with particular reference to computer graphics systems. There are, in fact, several types of data-structure, including tree, ring, plex and parallel structures. The tree and ring structures are particularly appropriate for display work, and will briefly be described here.

Tree structures are built with the aid of 'pointers'. 'Pointers' are cells in a data-structure containing the addresses of other cells. By linking a group of cells a simple 'tree' is created. (Figure 2.1).

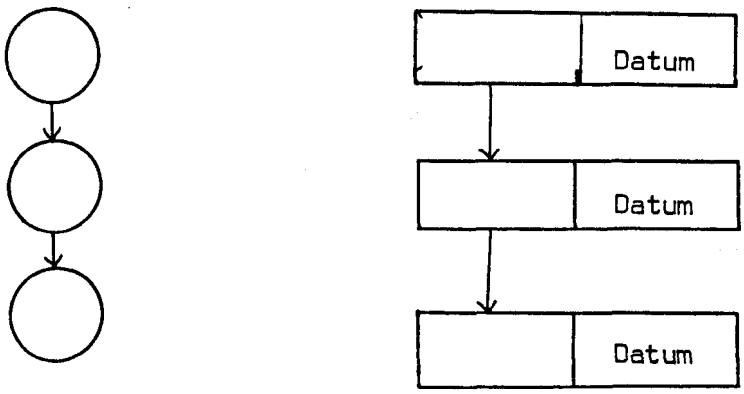


Figure 2.1

Figure 2.1 represents a linked list structure, in which each element in the list consists of a datum and a 'pointer' to the next element. The final 'pointer' in the list is set either to zero or to a null value.

Lists can be used to describe the attributes of items in one of two ways: either by enumerating all the attributes of a single item (Figure 2.2) or by enumerating all the items by attribute (Figure 2.3).

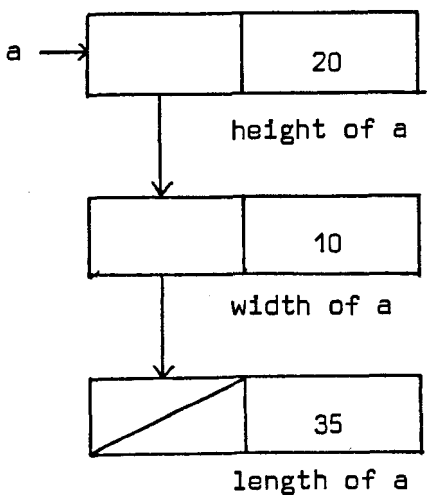


Figure 2.2.

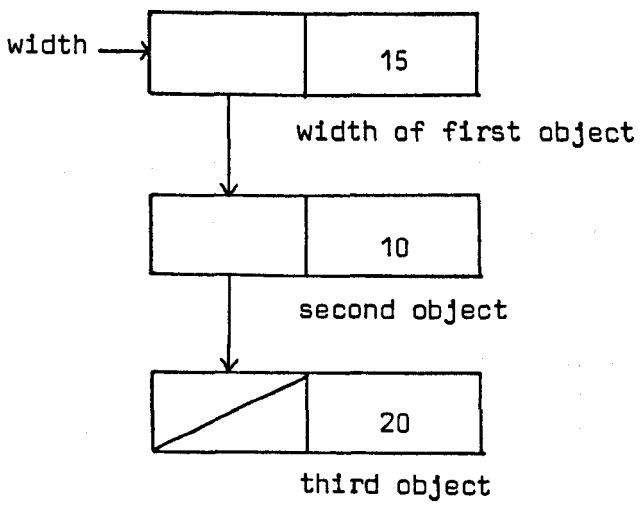


Figure 2.3.

It is also desirable to be able to describe relationships between one item and another: by, for example, specifying that one point is the nearest to another one. This can be achieved by using 'pointers' to indicate references, as shown in Figure 2.4.

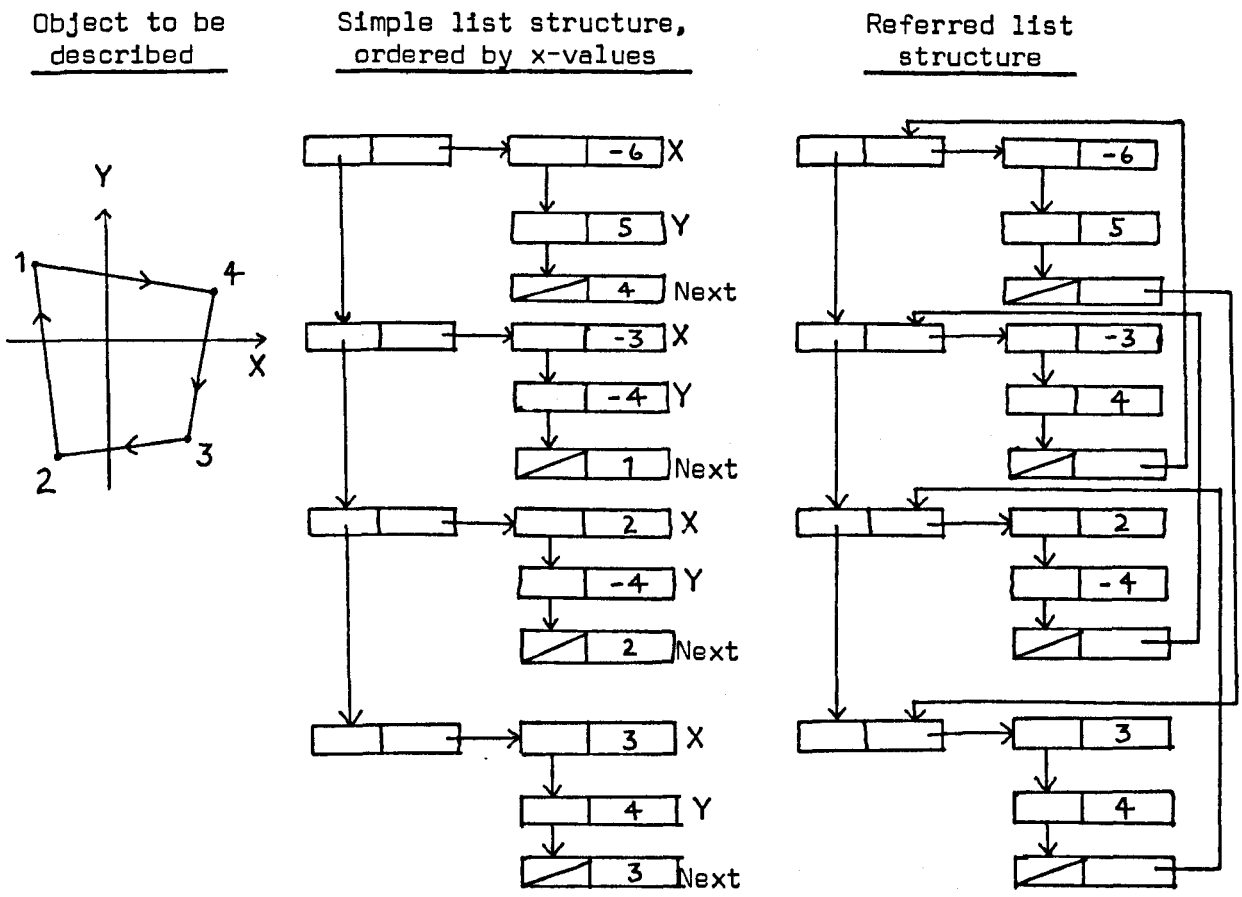


Figure 2.4

One operation that the above scheme does not permit is reverse referencing, that is, the capacity to trace from a referred element to the referencing element. This can be provided by the use of a system of two-way pointers but entails an unmanageable increase in the size of the structure. This difficulty can be overcome by the use of a ring structure, which defines

as a set, all the elements containing references to a particular element and links them to the referenced element. An additional advantage of this technique is that it makes possible the easy insertion of new items of information without involving major alterations in the structure.

2.3. Properties of an Interactive Graphics System

Before considering the overall facilities offered by a system of interactive computer graphics, it is necessary first to consider the individual properties of computer graphics and of interaction.

One of the principal aims of computer graphics is to replace numbers by pictures in such a way as to allow information to be presented in a more concise, compact and economic manner. In general a display presents a pictorial summary of information which may entail a loss of detail but gives a better overall view of objects or data shapes, trends and distributions.

Interactive facilities allow communication with a program during its run time. This means that information can be supplied, and decisions made, based on intermediate results, which may then be used to influence the rest of the computation. It is thus possible to experiment on-line, without having to wait for output to be returned from a batch run. Interaction can in fact range from simple question answering, to complete control of a computer program.

An interactive graphics package can be programmed to be controlled and wholly run from the display device using the keyboard together with available input devices. Alternatively if a large quantity of data is needed, other methods of input, such as cards or disc files can be used. In either case a program may include a checking procedure, where data is displayed and if necessary corrected, before computation continues. In the case of a large quantity of data it is sometimes useful to display it in

pictorial form either as a curve or surface or by drawing the region or object to be generated. Interactive graphics considerably ease the interpretation of 3-dimensional objects or surfaces, as they can be viewed in perspective, rotated about an axis or represented by a contour plot.

To summarise: the major advantages of interactive graphical methods over batch processing are:

(a) speed,

(b) intermediate graphical output is available,

(c) allows simple and efficient interactive control of input, calculations and output.

2.4. Applications of Interactive Computer Graphics

The volume and variety of applications of interactive computer graphics, both in research and commercial environments has increased considerably in the last few years. Some of the most important areas are:

Computer-aided design

Management Information Systems

Simulation

Process Control

Pattern Recognition

Computer Animation

Education

2.4.1. Computer-aided Design (CAD)

This is one of the largest areas of application of interactive computer graphics, involving many important industrial organisations. In the aircraft industry for example, the Lockheed Aircraft Corporation[10] uses an interactive graphics terminal for the numerical control of cutting machinery, and at McDonnell-Douglas[38] the technique is applied to the

design of aerodynamic three-dimensional shapes, to structural analysis, design drafting and also to numerical control. In the motor industry C.A.D. is also widespread being used in companies such as the Ford Motor Company and British Leyland. These and other applications are described in Computer Graphics in Management[19]. Another specific area of C.A.D. into which interactive graphics is making significant inroads, is the design of integrated circuits. Examples of this are given in Advanced Computer Graphics[46]. Other applications include the use of C.A.D. for modular building elevation design[12] and for the design of the sculptured surfaces of moulded products[53].

2.4.2. Simulation

Graphics terminals are increasingly in use as output devices for simulators. The 'Sketchpad' program (which is used to simulate the design of bridges) was a pioneer in this field. Displaying the structure of a bridge on a fluorescent screen, 'Sketchpad' allows the stress forces of any part of the structure to be calculated and permits the designer to modify or alter elements in the structure, and thus in the stress pattern, by using a light-pen. On-line problem solving such as the solution of differential equations, can be considered as a simulation problem, (see Computergraphical Description of the Dynamics of a Turbulence Model[46]) as can the modelling of chemical structures.

2.4.3. Process Control

Typical applications in process control are described in 'Man/machine Communication and Process Control' by D. E. Weisberg[59]. For example, he describes closed-loop processes used to monitor the past and future states of several variables. In such systems an operator can control

pumps and valves by pointing a light-pen at their symbolic representations on a screen.

2.4.4. Pattern Recognition

Interactive computer systems have been developed to optically scan film or paper documents for analysis and data reduction. Areas of application include on-line text reading and editing, and image and signal processing. An interactive computer-controlled scanning and display system is in operation at the IBM Research Laboratories in New York. A full description of this system can be found in 'Design of an Experimental Laboratory for Pattern Recognition and Signal Processing'[46].

2.4.5. Computer Animation

One of the attractive features of computer animation and computer generated films is their relative economy. The cost of corresponding hand-drawn films is at least twice as much in simple cases and in other cases animation would not be possible at all without a computer. Burtnyk and Wein[9] describe a system for the professional animator in the traditional film industry where the main application has been animation with free form drawn images.

2.4.6. Education

There have, as yet, been very few applications of interactive computer graphics in the field of education in the United Kingdom. In the United States, however, the concept of computer-assisted learning, which grew up in the 1960's, has expanded in recent years to include the use of interactive graphics. A typical system uses a visual display unit to display teaching text. The learner reads the text and is then questioned by the computer about what he has read. Answers are typed on a keyboard and the

learner is then fed further questions or data, depending on the answers given. A system using an interactive graphics terminal is in use as a teaching aid to undergraduate engineers at Queen Mary College, London. This, and other systems, are described in 'Computer Assisted Learning in the United Kingdom'[32] edited by Richard Hooper and Ingrid Toye. The 'state of the art' in the United States is presented in 'Computer-Assisted Instruction[2] edited by R. C. Atkinson and H. A. Wilson.

Only a few of the areas of application of interactive graphics are mentioned here. Those omitted range from medical research and underwater technology, to the Graphic Arts. In the future more economic machines and developments such as colour display, will no doubt further broaden the field of application.

3. INTERACTIVE GRAPHICS AT THE UNIVERSITY OF KEELE

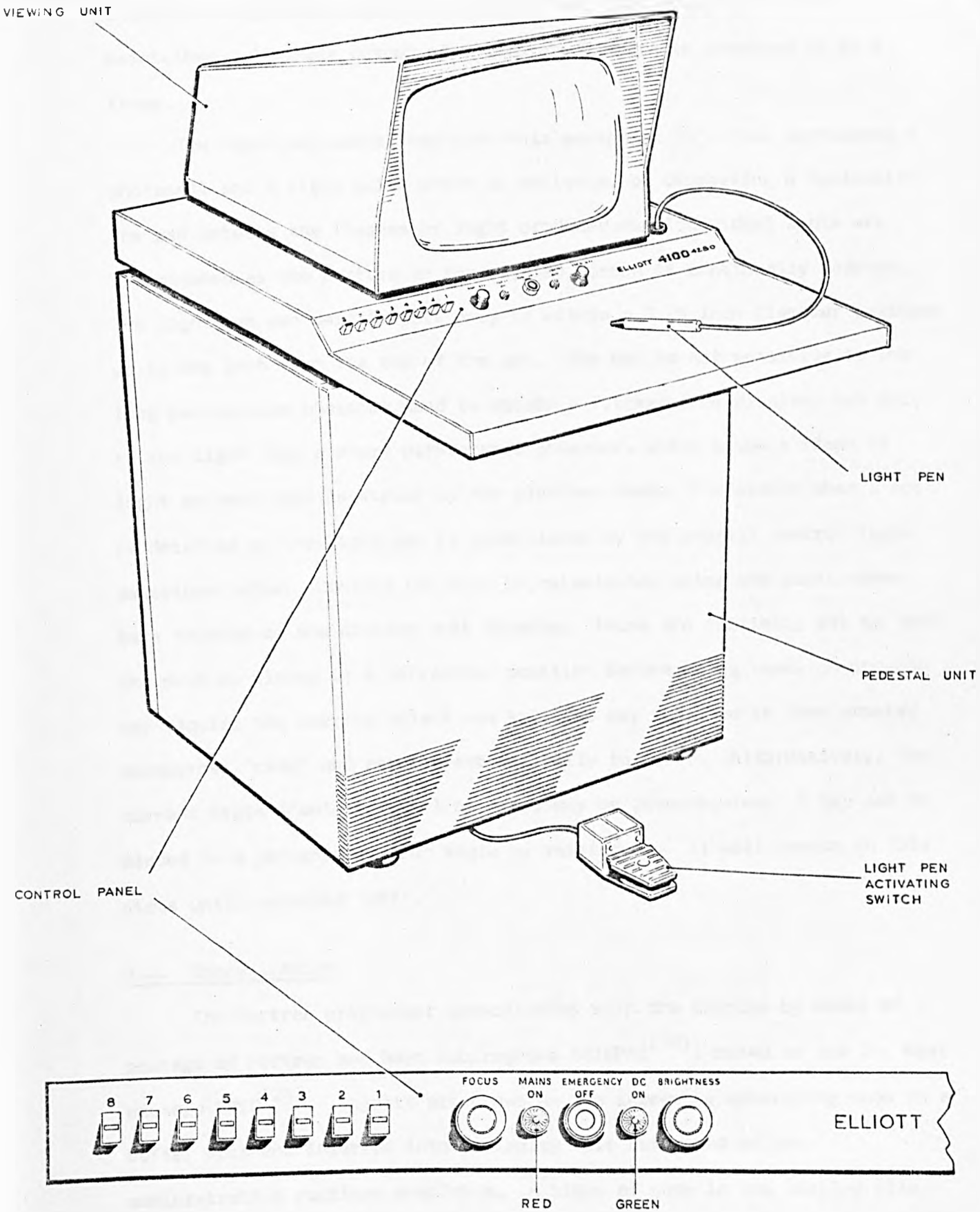
The programs described later were written for use on an ICL 4130 computer with a 4180 graphics display terminal. A brief description of the terminal and available input devices is given below together with an outline of how the system is used.

3.1. Description of the System

The Interactive Graphics system at Keele consists of a 4180 display terminal with associated keyboard, sense keys and a light-pen. The viewing unit, symbol generator and their power supplies, together with the light-pen, its power supply and the eight sense keys, are housed in a display console (Figure 3.1). The display controller is normally located elsewhere. The keyboard usually used is an on-line terminal placed near the graphic display unit, but for the system used in the development of the following programs, the operator's console was used as the keyboard.

A picture is produced on the display screen in response to information held in coded form in the main store of the 4130 computer. The picture, displayed on the cathode ray tube, is regenerated approximately ten times every second and is maintained, flicker free, by the use of a long persistence phosphor. The coded information is automatically extracted from the computer store as it is required and central processor time is minimised since its operation continues in parallel with picture generation.

The picture is built up as a set of spots and lines. A square array of possible spot positions, arranged in a grid of 1024 rows and 1024 columns, gives a resolution of about 0.01 inches on a 10 inch square picture. Any spot once illuminated, will remain bright for approximately 100 milliseconds. The entire picture must therefore be redrawn at



DISPLAY UNIT & CONTROLS

Figure 3.1

intervals no greater than this, if a flicker free display is to be maintained. A single output of a complete picture is referred to as a frame.

The light-pen associated with this equipment is a tube containing a photocell and a light guide which is activated by depressing a footswitch. The pen detects the flashes of light produced when individual spots are illuminated as the picture on the display screen is continually redrawn. The light pen can detect spots only to within a 0.25 inch diameter cylinder up to one inch from the tip of the pen. The pen is not sensitive to the long persistence phosphor used to obtain a flicker free display, but only to the light from a short persistence phosphor, which gives a flash of light as each spot is struck by the electron beam. The action when a spot is detected by the light-pen is conditioned by the overall control logic described below. Control can also be established using the eight sense keys mounted on the display unit console. These are initially set to 'off' and must be placed in a horizontal position before being used. A program may require the user to select one key; the key selected is then pressed downwards, 'read' and returns automatically to 'off'. Alternatively, the current state ('on' or 'off') of a key may be investigated. A key can be placed in a permanently 'on' state by raising it. It will remain in this state until switched 'off'.

3.2. Communication

The Fortran programmer communicates with the display by means of a package of Fortran and Neat subprograms (DISPAC[18]) based on the ICL Neat package FRED[24]. Objects are drawn on the screen by generating code in a buffer area and inserted into a display file using one of the administrative routines available. A block of code in the display file created in this way is referred to as an item. Items may or may not be

assigned an identifying number. It is therefore possible to assemble a picture from separate items, which can then be manipulated individually. The subprograms in DISPAC fall into four categories: administrative routines; code generating routines; control routines; and graph drawing routines.

1. Administrative Routines

Before generating code for an item, areas of store must be allocated for the code buffer and for the display file. Certain setting-up routines must also be activated. These tasks, together with the organisation of the items in the display file, are performed by the administrative routines. As an example, GSTART must be called before any other DISPAC routine. GSTART sets the current units needed to define points on the screen to either inches or centimetres, opens the display file, sets up default values for such things as character size and brightness, and sets the maximum number of items that can be displayed simultaneously. The code in a buffer may be numbered, inserted into a display file and thus displayed on the screen by a call to GINSRT. Other routines in this category are used to set up empty code buffers, to delete, replace, rename and move items; and to change parameter settings for character size, brightness and edge status in a particular item.

2. Code Generating Routines

These routines can be considered the basic building blocks of a picture. They are used for absolute and relative beam positioning, line and character drawing, and text output. As an example; GPOINT(X,Y) generates an absolute beam position at the point (X,Y) where X and Y are in current units as defined by GSTART. A routine is also available to change parameter settings. This may be useful when different settings,

for example, character size, are required in different parts of a picture, and may be achieved by including the parameter setting command GSIZE in the appropriate item. The graph drawing routines also fall into this category, but they will be described in more detail later.

3. Control Routines

The user is able to interact with a program while it is running using the light-pen, sense keys and the alphanumeric keyboard. The keyboard may be used with the usual READ statements, although numerous special routines are also available to give more flexibility. A routine 'GACT' is used as the main controlling routine and incorporates all three means of interaction. It is used in the form I = GACT(NOPT,NRES) where NOPT is set by the user as follows:

NOPT = 1 allows a sense key to be depressed.

= 2 allows NRES frames to be displayed.

= 4 allows a light-pen hit.

= 8 allows a character to be typed.

Combinations of options are obtained by setting NOPT to the sum of values for each option required. NRES indicates the first event to occur.

NRES is set to 1 if a sense key is depressed.

2 for frame count expiring.

3 for a light-pen hit.

4 for a character input.

I takes the sense key number if NRES is 1, zero if NRES is 2, the item number detected if NRES is 3, and the machine code equivalent of the character input if NRES is 4. The light-pen can also be used with a tracking cross to locate points or items. The sense keys may be 'read' using GKEYS, in the form K = GKEYS(I,J) where K is the binary value of

keys I to J inclusive. ('Set' implies one here, and unset - zero). They may also be 'read' using the logical function GSWITCH(I), which is TRUE if key I is set and FALSE if it is unset.

4. Graph-drawing Routines

To draw a graph it is necessary to set up a data area in which points can be defined in terms of data units. The area is defined using GDAREA which specifies the dimension in current units (inches or centimetres). The datum and scale can be either specified by the programmer using GDATUM, GXSCAL and GYSCAL or be calculated from a set of data points using GSCALX and GSCALY. Within the data area the plot may be titled and axes drawn with tick marks and labels as required. Lines are drawn within the data area using the routine GLINE, which joins a set of points given in data units. The routine GCURV is used to draw curves which are fitted to a set of points using a least-squares procedure based on cubic splines. The curves are drawn as a set of straight line segments, evaluated by interpolating the splines at prescribed intervals along the abscissa.

An additional facility enables a 'hard copy' of the display file to be obtained on a digital plotter simply by calling the NEAT subprogram GPLOT.

3.3. Job Control on the 4130

A standard Fortran program, without the use of disc files or program segmentation, needs the following control cards:

```
&JOB;<user identifier>;<job name>;
&FORTRAN;
(&LIST;)     optional
  .
  .          program cards
  .
&RUN;
  .
  .          data cards
  .
&END;
```

The programs described in Chapter 6 are segmented and held in disc files some of which are compiled and some of which are not.

DISPAC (the graphics routines) is held in compiled form, on a systems disc which has the volume number zero. It is accessed with the control card:

```
&ASSIGN;3;DC;0; <any routine>, DISPAC;
```

Since only one library of compiled routines can be assigned to channel three and hence accessed automatically, all other compiled routines must be fetched individually when the program is loaded. An input channel is assigned to these routines, which are held on the user disc, volume number 91, using the control card:

```
&ASSIGN;101;DC;91; <any routine>,<user identifier>;
```

The programs also contain a number of uncompiled routines which in each case are held in one file, together with the 'FETCH' statements.

The files containing uncompiled routines are also held on user disc 91 and each can be assigned to channel one using this card:

```
&ASSIGN;1;DC;91; <file name>,<user identifier>;
```

The '&ASSIGN' cards described above must appear immediately after the 'job' card, but may be in any order.

Each program must also include an 'OPTIONS' card, having the parameter 'FIORD', to show that files held on disc will be accessed, and the parameter DSEG to indicate a segmented program. This card, located immediately before the &FORTRAN card, will appear thus:

```
&OPTIONS;FIORD;DSEG;
```

For each program, the text file (PGSOPE) is held on disc as a V-file or data file. It is designated channel 70, and accessed with the card:

```
&ASSIGN;70;DV;3;PGSOPE
```

which is placed with any other V-file ASSIGN cards before the &RUN card.

Assuming that all files (other than DISPAC) are held under the user identifier CC08 the job control cards necessary to run the program GIDODE will take the following form:

```
&JOB;CC08;GIDODE;
&ASSIGN;3;DC;0;GSTART,DISPAC;
&ASSIGN;101;DC;91;GEQNTP,CC08;
&ASSIGN;1;DC;91;GIDODE,CC08;
&OPTIONS;FIORD;DSEG;
&FORTRAN;
&LIST;
  .
  .  user routines if necessary
  .
&ASSIGN;70;DV;3;PGSOPE
&RUN;
  .
  .  data if necessary
  .
&END;
```

The &LIST; card only lists routines provided on cards, that is the user routines; and not those held on disc.

For GIDPDE the control cards are:

```
&JOB;CC08;GIDPDE;
&ASSIGN;3;DC;0;GSTART,DISPAC;
&ASSIGN;101;DC;91;GEQNTP,CC08;
&ASSIGN;1;DC;91;GIDPDE,CC08;
&OPTIONS;FIORD;DSEG;
&FORTRAN;
&LIST;
  .
  .  user routines if necessary
  .
&ASSIGN;70;DV;3;PGSOPE;
&RUN;
  .
  .  data if necessary
  .
&END;
```

For the program FESOLV it is possible to store data in a disc file. If this option is required during a program run, it is necessary to assign channel 80 to a disc file. The file name should include the user identifier, and for simplicity may also incorporate the name of the problem identifier.

The program FESOLV is loaded using the following cards:

```
&JOB;CC08;FESOLV;
&ASSIGN;3;DC;0;GSTART,DISPAC;
&ASSIGN;101;DC;91;TRNSFM,CC08;
&ASSIGN;1;DC;91;FESOLV,CC08;
&OPTIONS;FIORD;DSEG;
&FORTRAN;
&ASSIGN;80;DV;3;PGSOPE;
&ASSIGN;70;DV;3;CC08  <problem identifier>;
&RUN;
  .
  .   data if necessary
  .
&END;
```

4. MATHEMATICAL FORMULATION AND TECHNIQUES

This chapter begins with a discussion of the types of equations that are included in a set of interactive graphics programs, described in Chapter 6, for solving differential equations. This is followed by the selection of appropriate numerical solution techniques. The ancillary activities, curve fitting, solution of linear algebraic equations, representation of 3-dimensional surfaces and the input of algebraic expressions are also discussed.

4.1. Mathematical Formulation of the Problem

It is now necessary to decide what types of ordinary and partial differential equations may sensibly be included in an interactive graphics system. There are various issues to be considered when making this decision.

1. The equations must describe realistic non-trivial problems, but, since the main purpose is to study the methodology and to illustrate the advantages of interactive graphics, reliable methods of solution must be available.

2. It is obviously more efficient to use some general form of equation which will cover a range of problems.

3. The solution of the problem should be simplified in some way by using interactive graphics.

4. The different types of equations can be chosen so that various characteristics of an interactive graphics system can be examined and illustrated.

For ordinary differential equations satisfying most of these criteria is not difficult. Consider the general system of N first order equations

$$\frac{dy_i}{dx} = f_i\,(x, y_1, y_2, \ldots y_N) \quad i = 1, 2 \ldots N \qquad 4.1$$

where the f_i's are known functions of $x, y_1, y_2, \ldots y_N$. This covers a large range of non-trivial problems including higher order systems which can easily be converted into this first-order form. Many programs are currently available to solve boundary-value and initial-value problems expressed in this form: for example, programs in the Numerical Algorithms Group Library[44] are available to solve boundary-value problems and both 'stiff' and 'non-stiff' initial-value problems. Stiff systems of equations occur in many fields[8], for example, chemical reactions, electrical circuits and control problems and are characterised by having rapidly decaying transient solutions. The difficulties arising with this type of problem are discussed when considering computational methods of solution.

For partial differential equations we want to illustrate the advantages of interactive graphics using the two major solution techniques: finite differences and finite elements. The finite element method will be used to solve elliptic equations since the method has only been extended to time dependent problems relatively recently[66]. A fairly comprehensive theoretical basis is available for elliptic equations[54],[65],[61] but very few convenient variational principles, on which the finite element method is based, are available which describe time dependent processes[42]. Time dependent problems have the additional disadvantage that for a two-dimensional problem three-dimensional elements are required. To avoid the storage and organisational difficulties that arise when solving large or complex systems only a very limited range of problems will be considered here.

The general form used is

$$-\nabla^2 u + gu = f \qquad 4.2$$

where $u = u(x,y)$, $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$, and g and f are known constants. The region of solution is a plane domain Ω bounded by a continuous closed curve Γ. The boundary conditions are

$$u = a(x,y) \quad \text{on } \Gamma_1$$

and

$$u_n - pu = q \quad \text{on } \Gamma_2$$

where p, q are known constants and u_n is the normal derivative of u at the boundary.

A finite difference method is used to solve parabolic partial differential equations in order to incorporate time dependence. In the two-dimensional case this is easily adapted to include the steady state elliptic problem.

Both one and two dimensional parabolic equations are included since quite different display problems arise in each case. In the one-dimensional case many solution curves can be displayed simultaneously for comparison whereas in two dimensions the problem is in displaying a three-dimensional surface on a plane. The general form for the two-dimensional problem does not include the mixed derivative term since this complicates the computation considerably and could in any case be removed from an equation by transforming co-ordinates[64]. Except for this minor restriction a wide range of problems and boundary conditions are covered.

In one dimension the general form is

$$u_t = A_1 u_{xx} + A_2 u_x + A_3 u + A_4 \qquad 4.3$$

within the open rectangle $[x_{min} \leq x \leq x_{max}] \times [t \geq t_{min}]$.

Initial conditions are $u_o = F(x)$

and boundary conditions $u_b = g(t)$

$$\text{or} \quad \frac{\partial u_b}{\partial x} \pm h(u_b - v) = 0$$

where $u = u(x,t)$, A_i, $i = 1, 2, 3$ and 4, are functions of x and t and u_b is $u(x_{min},t)$ or $u(x_{max},t)$.

The general form for two dimensional parabolic equations is

$$Bu_t = A_1u_{xx} + A_2u_{yy} + A_3u_x + A_4u_y + A_5u + A_6 \qquad 4.4$$

within the region $[x_{min} \leq x \leq x_{max}] \times [y_{min} \leq y \leq y_{max}] \times [t \geq t_{min}]$.

Initial conditions are $u_o = F(x,y)$

and boundary conditions $u_b = g(x,y,t)$

$$\text{or} \quad \frac{\partial u_b}{\partial n} \pm h(u_b - v) = 0$$

where $B = 1$, $u = u(x,y,t)$, A_i, $i = 1, 2 \ldots 6$ are functions of x, y and t, u_b is a boundary value of u and $\frac{\partial u_b}{\partial n}$ is the derivative of u normal to the boundary at $u = u_b$.

For elliptic equations in two dimensions the general form is equation 4.4 with $B = 0$. The region of solution is $[x_{min} \leq x \leq x_{max}] \times [y_{min} \leq y \leq y_{max}]$ with boundary conditions $u_b = g(x,y)$

$$\text{or} \quad \frac{\partial u_b}{\partial n} \pm h(u_b - v) = 0.$$

4.2. Computational Methods of Solution

At this stage the general forms of equations have been chosen for which reliable methods of solution are available. We now need to select the particular method to be used for each of the equations 4.1 - 4.4. Other points to be considered when making this choice are that the methods should be well tested, results should be produced in a displayable form, and the implementation of a method should allow intermediate decisions, such as the choice of step length, to be made interactively by the user, if required.

For the solution of the ordinary differential equations described by 4.1 the original methodsused were those available in the N.A.G. library[44] for the solution of two-point boundary-value problems (D02ADF), initial value problems (D02ABF) and stiff problems (D02AEF). The methods used by these routines are Merson's method with Newton iteration[30] for boundary value problems, Merson's method for non-stiff initial value systems[40], and Gear's method for stiff systems[25]. However, the N.A.G. routine D02AEF for stiff equations proved unsatisfactory. On some occasions it failed to converge or output an error message (although convergence could be obtained when the same problem was run with, allegedly the same N.A.G. library routine via the 7020 link to the Manchester Regional computer) and in addition the routine could not be used when solving one equation only. Further reference to the unreliability of D02AEF is made in a report by J. A. I. Craigie in August 1975[14]. The routine D02AEF was therefore replaced by DIFSUB, the routine originally published by Gear in 1968[26]. This proved much more reliable, and since it deals with non-stiff as well as stiff equations the routine D02ABF was also replaced. In a paper 'Comparing Numerical Methods for Ordinary Differential Equations'[33]

Hull et. al. consider multi-step methods to be generally superior to single step methods for the solution of initial-value problems and in fact recommend the use of either Krogh's or Gear's implementation of Adams' method. For clarity Gear's implementation of Adams' method for non-stiff equations will be called Adams' method and that for stiff equations will be called Gear's method.

As stated earlier the finite element method is used for only a small range of problems. It is not clear whether it is more efficient to subdivide the region of solution into triangles or quadrilaterals; triangles are obviously a better approximation at curved boundaries although there are advantages in using quadrilaterals, (especially rectangles) in the interior. Rather than mixing element shapes (which may be the best choice) triangles are chosen with the simplest and most basic of trial functions, a linear function within each triangle.

The finite difference methods used to solve the one (4.3) and two (4.4) dimensional parabolic equations are the Crank-Nicolson scheme[16] and the Peaceman-Rachford Alternating Direction method[48] respectively. These are well-tested implicit methods and unlike explicit methods converge for large time steps. Other methods such as The Hopscotch Method[28] were considered, however, since one of the advantages of the interactive graphics system is that the user is able to view the development through time at each point the economic advantages of 'Hopscotch' would be lost.

4.3. Curve Fitting and Linear Algebraic Equations

In this section the numerical methods used, during the ancillary activities of curve fitting and solving linear equations, will be discussed.

4.3.1. Curve Fitting

Curve fitting is necessary when plotting solutions or errors and when drawing contours. In the programs to be described in Chapter 6 the initial method used was to firstly fit polynomials to the data points using a least squares method. The best polynomial fit was then interpolated at equal intervals and these points were joined together with straight lines. The method however, had certain disadvantages. The principal one being ill-conditioning. This arises because the powers of x (the independent variable) are not orthogonal and can be overcome by using orthogonal polynomials. However, previous experience suggested the use of cubic splines[1] which have the additional advantage, when used in parameter form, of not requiring the function to be single-valued.

4.3.2. Linear Algebraic Equations

The solution of linear algebraic equations arises during the finite difference methods for solving partial differential equations, in the cubic spline method of curve fitting, and in the final stage of the finite element method. If we consider the system of equations $A\underline{x} = B$, then in the first two cases the matrix A is tri-diagonal and satisfies the necessary criteria for solution using the tri-diagonal algorithm based on the Gaussian elimination method (see pp 586-588 of (64)). This method is used since it requires less storage than the alternative method for sparse matrices, namely, iterative techniques. The matrix K arising in the discrete finite element system $KQ = F$ (see Appendix B) is symmetric, large, sparse and banded. For this type of system iterative methods are generally preferred to direct methods such as Gaussian elimination, however in this case it is desirable not to store the complete matrix K. The method chosen is one devised by

A. Jennings[36] and is an adaptation of the Choleski method for symmetric positive definite matrices with variable band width. The only elements stored are those from the first non-zero element up to the diagonal element of each row.

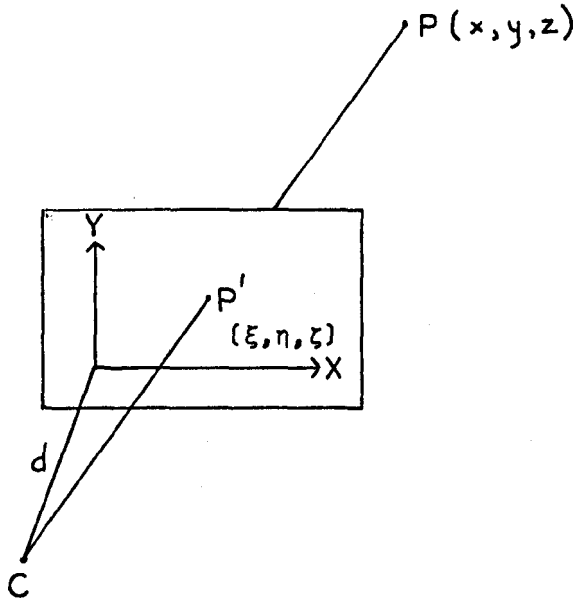
4.4. Representation of a 3-Dimensional Surface

When solving any of the two-dimensional equations described above the problem arises as to how best to display a function of two variables so that the viewer can get both a qualitative and a quantitative description of the function. Two methods of approach will be considered, one is to display the whole surface as a projection on to a plane and if necessary allow it to be looked at from different viewpoints, and the other is to represent the surface by a set of curves of equal value, that is, by contours. Whereas contour plots are most convenient if numerical information is to be extracted from the figure, a planar projection enables the overall properties of the surface to be visualised more easily.

4.4.1. Projection

Various methods are currently available to give the illusion of depth in a three-dimensional object drawn in two dimensions. These include kinetic depth effects, achieved by either continuous projection of a rotating three-dimensional object[39] or a moving viewer, as in the case of Sutherland's head-mounted displays[56], depth cueing using, for example, shaded pictures[57] and stereo pair displays[45]. However, the simplest method, requiring no additional hardware is that of projection on to a plane[37]. Perspective planar projection gives the best representation of objects as viewed by the naked eye and is used for all the 3-dimensional surfaces displayed in the following programs.

This type of projection maps an arbitrary point P onto a point P′, on a plane π, such that all lines PP′ intersect at a common point C. The point C is called the centre of projection or the viewpoint. The plane π is the plane of projection and is orientated so that its normal is parallel to the line of sight.



The projection of the point P(x,y,z) can be found from the following formulae. These are derived by Kubert et. al.[37]. Given the centre of projection, (c_x,c_y,c_z), the direction of sight (cos α, cos β, cos γ), and the distance d of the plane of projection, then

1. $q_x = c_x + d(\cos\alpha)$; $q_y = c_y + d(\cos\beta)$; $q_z = c_z + d(\cos\gamma)$
2. $K = d/[(x - c_x)\cos\alpha + (y - c_y)\cos\beta + (z - c_z)\cos\gamma]$
3. $\xi = c_x + K(x - c_x)$, $\eta = c_y + K(y - c_y)$, $\zeta = c_z + K(z - c_z)$

and 4. $X = [(\xi - q_x)\cos\beta - (\eta - q_y)\cos\alpha]/\sin\gamma$, $Y = (\zeta - q_z)/\sin\gamma$

or if $\sin\gamma = 0$

4. $X = [-(\xi - q_x)\cos\gamma + (\zeta - q_z)\cos\alpha]/\sin\beta$, $Y = (\eta - q_y)/\sin\beta$

This method has one major drawback in that the parts of the object or surface that are not visible from the viewpoint are also projected onto the plane of projection. This property may be useful in some applications but often makes the picture difficult to interpret. It would be desirable to plot only that part of the surface that is visible from a particular viewpoint. This can be achieved by the removal of the hidden lines or surfaces.

A great deal of effort has been applied to solving the hidden-line problem as considerable computation is required to decide which lines or surfaces are hidden. The first practical solution is given by Roberts[49] who inspired many workers in the field. His algorithm has been followed by many others[57,63] including Warwick[58]. The hidden lines have not been removed from the projections used in the programs described later as the improvement in picture quality did not justify the increased storage and time requirements.

4.4.2. Contour Plotting

The problem of contour plotting has been approached by workers in various fields including mathematics, engineering and geography. In 1969 Morse[43] discussed generalised techniques for simplifying the solution of problems relating to contour maps. Automatic contouring programs have been available since the early 1960's and in [13] methods for contour mapping given data points on a rectangular grid are discussed and a new method described. In a later algorithm [15] a method is presented to plot contours for functions specified at nodal points of an irregular mesh based on an

arbitrary two parameter co-ordinate system. More recently, McLain[41] describes an algorithm for drawing contours given an arbitrary set of data points.

In the program described later, contour plotting is available only when the finite element method is used. Data points are then given at the nodes of triangles and are transferred to a rectangular grid using the weight least squares routine described by McLain. Contours are then evaluated from these data points using Crane's algorithm[15].

4.5. Input of Algebraic Equations

It is often necessary when solving mathematical problems to supply functions $y_j = f_j(x_i)$ for $i = 1, \ldots n$ and $j = 1, \ldots m$, which can be used to evaluate the dependent variables y_j for particular values of the independent variables x_i. It is possible to supply required functions either as Fortran subroutines or function segments prepared on cards before the program is run by using default functions, or by inputting them at run time via the keyboard.

Fortran subroutines or function segments supplied with the job control cards will be referred to as 'User Routines'. Their position relative to the job control cards is shown in Chapter 3.3. It is sometimes necessary to supply more than one user routine for a particular problem, so, in order to distinguish between them the name of each routine is uniquely defined by the calling program. For example, the program to be described in Chapter 6 to solve parabolic partial differential equations in two dimensions requires that, if the initial conditions are supplied by a user routine, then it must be called UINIT. Alternatively, a routine used to evaluate a certain type of boundary condition must be called BNDZT.

To illustrate the form of a typical user routine consider the problem of supplying the initial conditions necessary to solve a two-dimensional parabolic partial differential equation (see equation 4.4). The initial conditions are of the form u = F(x,y) and as an example we shall consider $u = 1 + x + 2y^2$. The routine must be called UINIT and will take the form

```
FUNCTION UINIT (L,NF,X,F)
DIMENSION X(2), F(NF)
UINIT = < F(1) × f1 + F(2) × f2 + ... F(NF) × fNF>
          or <  g  >
RETURN
END
```

where expressions shown <·> must be replaced appropriately for a particular function.

f_1, f_2 ... f_{NF}, g — define particular functions of the independent variables x and y.

X — is a real array, length 2. X(1) holds the particular value of x, and X(2) the particular value of y for which UINIT is to be evaluated.

F — is a real array, length NF, containing the constants to be used in the definition of UINIT.

NF — is an integer and defines the size of the array F.

L — will hold the value 1 for a one-dimensional problem and 2 for a two-dimensional problem. It need not be used but must be included in the argument so that the user function takes the same form as the default function.

The constants held in NF and the array F are either input as data on cards or via the keyboard or alternatively need not be used. F is not used to store constants, the expression g will contain actual values for the co-efficients of each term. It may at first seem rather unnecessary to

allow the constants F_i i = 1, 2 ... NF to be supplied separately. This is, however, sometimes a very useful facility since it allows the same function segment to be used for separate problems merely by redefining the constants. For our example $u = 1 + x + 2y^2$ the routine can take one of the following forms

(1)
```
FUNCTION UINIT (L,NF,X,F)
DIMENSION X(2), F(NF)
UINIT = F(1) + F(2) * X(1) + F(3) * X(2) * X(2)
RETURN
END
```

(2)
```
FUNCTION UINIT (L,NF,X,F)
DIMENSION X(2), F(NF)
UINIT = 1.0 + X(1) + 2.0 * X(2) * X(2)
RETURN
END
```

In the first case the values NF = 3, F(1) = 1.0, F(2) = 1.0 and F(3) = 2.0 are supplied separately as data.

Two other ways of supplying functions exist. The most trivial is to input coefficients to be used with a default function. The default function for the initial conditions of a two dimension parabolic partial differential equation is $u = c_1 + c_2x + c_3y$ where the values c_1,c_2 and c_3 are either typed on the keyboard or input on data cards.

The third method is to input the complete function at run time via the keyboard. This makes full use of the interactive facilities available to an operator. A subroutine STATE is used to 'read' a series of arithmetic expressions typed on the teletype keyboard while a program is active. The expressions are then evaluated when necessary during a program run using a routine UNSTAT. The routine STATE decodes an expression into Reverse Polish Notation form[29] before storing it in an array. This array can then be accessed by UNSTAT to evaluate the reverse polish form for particular values of the independent variables.

To type in expressions at run time it is necessary to define them in the form

Y = F(X1,X2 ... XM)

where the names X<i> will represent particular variables depending on the application. The implementation of this facility will be described in more detail in Chapter 6.

5. DESIGN CRITERIA

In Chapter 2 some of the most common areas of application of interactive computer graphics were described. This present chapter will be devoted to a closer examination of the mathematical problem solving process. Both computer-aided-design and simulation fall within this category in which the problem or objective is to find the solution or solutions of a mathematical model of some process (physical, economic or any other).

In the first part of the chapter the general design and implementation of an interactive computer graphics system in a scientific problem environment is discussed. This is followed by some further development of these ideas related to the specific area of differential equations.

5.1. Interactive Computer Graphics in Mathematical Problem Solving

A computer program written to solve a mathematical problem using interactive computer graphics is similar in many ways to any ordinary problem orientated computer program. It is designed to solve a range of specific problems using parameters supplied by the user. The problem is defined by the data supplied, some analysis is carried out and the results are returned to the operator. The main difference when using interactive computer graphics is that the program is run in real time, that is, input parameters are supplied and the output is displayed while the program is active. In addition, the operator can control the order in which segments of program are run and can interrupt the running at points prescribed within the program to select a route through a program from a list of programmed alternatives.

5.1.1. General Facilities

An interactive graphics system of programs written to solve

mathematical problems can be expected to have a number of problem-independent facilities. Some of the most obvious and useful facilities in this category will be described here. Other more problem-orientated facilities will be discussed in the next section.

(a) Program initiation

In order to access an interactive graphics system it may be desirable to simply sit at the display terminal and type the necessary commands on the teletype keyboard. This can be particularly useful if only a small amount of data is required for the problem and hence there is no need to prepare either control or data cards. If, however, a large amount of data is required it may be easier to prepare these beforehand, and to input both program control statements and data via a card reader (or some other input peripheral). The programs described in Chapter 6 for solving differential equations are initiated using job control cards (see Chapter 4.4) which are input via the card reader. The programs require a dedicated machine and hence the card reader is always available.

(b) Problem definition

Once an appropriate graphics program has been loaded one of the first tasks is to define the specific problem to be solved. There are two ways of achieving this, some combination of which can be made available in a particular program.

(i) Definitions and problem parameters are provided on-line at prescribed points in a program, using one or a combination of the display unit's input devices (light pen, sense keys or keyboard). For example, questions may be displayed on the screen together with instructions to the user to select a sense key or to type data on the keyboard. Alternatively, a user may be required to select from a list, displayed on the screen,

using the light pen to point at the chosen option. The most appropriate method at a particular stage in a program will depend on the type of information to be supplied. In the program written to solve either elliptic or parabolic partial differential equations the user must indicate in which category (elliptic or parabolic) his problem falls. An operationally simple method of achieving this is to display the alternatives, each with an associated sense key number, and to request the user to press the appropriate key. In general, the most efficient method of providing numerical data on-line is via the keyboard.

(ii) Data is prepared on cards and input with the control cards via the card reader. This is a particularly useful way of providing large quantities of numerical data.

The program to solve elliptic partial differential equations using the finite element method allows on-line data input, input from cards and a combination of these two methods. When defining a complex region shape, prepared data cards reduce the possibility of error as well as eliminating the computer time that would otherwise be used to type in data at run time. However, a simple region shape, such as a square, can be quickly defined on-line by typing the co-ordinates of each corner on the keyboard at the time requested. Certain choices, such as the choice of the input device to be used to define a region shape can only be made on-line. The user must select the device from a displayed numbered list by pressing its associated sense key number.

(c) Data validation

After the input of data it is necessary to check for errors and to make any corrections before the analysis begins. Information to be validated can be displayed either as lists of decisions made, with numerical values

assigned to the parameters defined, or in some graphical form designed to simplify error detection.

(d) Choice of solution technique and computational parameters

A choice of solution technique cannot always be made available. However, it will generally be necessary, if a numerical method of solution is being used, to define certain computational parameters such as step sizes, acceleration parameters and accuracy requirements. In the following programs, it has proved desirable to make the selection of solution technique completely interactive by requiring the user to select from a displayed list using the sense keys. In all cases, computational parameters can also be supplied on-line, although it is sometimes possible to provide them on data cards if desired. The programs also allow the user to change these parameters at prescribed points.

(e) Monitoring of solutions and errors

During the computation it is generally desirable to display some summary of the 'solution and error so far'. It is, of course, at this stage that many of the advantages of graphical representation become apparent. If only a small quantity of information is to be displayed it can be presented either numerically or graphically, but usually, in the case of complex problems the amount of information to be conveyed is considerable, and hence graphical representation in a compact form becomes important. During the monitoring it must be possible to adopt a change in strategy, by, for example, changing parameters before continuing or restarting the solution stage.

(f) Storage of solutions

A facility for storing intermediate or final solutions in some way is often needed. A solution may be required for the duration of the program

so that it can be compared to other solutions or it may be necessary to make a permanent copy of solutions on backing store so that further analysis can be carried out at a later stage or using a different program. Both temporary and permanent storage is available, where appropriate, in the programs to be described.

(g) Hard copies of the display file

If the necessary hardware is available, a user may make a hard copy of the display file at any prescribed stage in a program. This facility is provided at a number of points in the following programs.

(h) Alternative forms of output

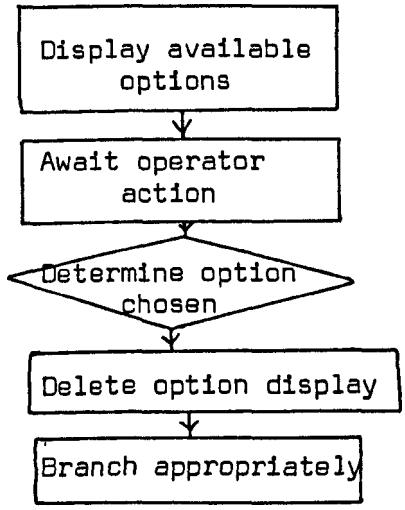
A number of alternative forms of output, both on the lineprinter and the display screen can be provided in an interactive graphics program. On the lineprinter the alternatives provided in all the programs described in Chapter 6 range from 'none' to complete problem definition, intermediate solutions and final solutions and errors. On the display screen the form of the output can be selected by the user from alternatives including numerical lists, individual graphs, multiple graphs and the various methods of displaying 3-dimensional figures described in Chapter 4.4.

The facilities described here may be available for only some of the time that a program is active, but will form part of an overall control network depending on a particular program structure.

5.1.2. Program Structure and Control

Interactive graphics programs exhibit certain structural features which distinguish them from other programming systems. These include the following.

(a) A large degree of choice can be made available to the operator at each decision point in the program. A typical structure of an interactive section of a graphics program is



It may be necessary in some cases to allow a branch back to the "Await Operator Action" stage (before deleting the display of available options) if more than one option can be selected at one decision point.

(b) A particular segment of program code may be re-used many times and in many different circumstances. This and the feature described in (a) make it desirable to divide a program into small but distinct logical segments. Care must be exercised not to take the modular form to extremes since both storage overheads and response times may suffer. A satisfactory attainment of low response times while maintaining program simplicity calls for considerable thought into the modularity of a program.

(c) The real-time environment calls for rapid responses to operator action. A user who has perhaps spent up to two hours in running a project at the display terminal will not be very tolerant of lengthy response times.

Control of the flow through an interactive graphics program can be exercised in different ways using the function keys, light pen or the keyboard. The relative merits of each depend to some extent on the form of the decision being made. For example, the light pen may be used as the major controlling device when selecting from a menu of displayed options particularly if the length of the menu is greater than the number of sense

keys. However, if only a small number of options are available the choice can be more accurately made if each option has an associated sense key number. If possible certain program standards can be adopted such as Key 1 to 'continue' and Key 8 to 'exit'.

It is usually desirable to allow the operator as much control over the program being run as circumstances permit. It should be possible for any logical sequence of operations to be selected and for processing to be interrupted at any stage in the program. A great deal of flexibility in the choice of route through a program is called for, so that, for example, an operator can return to earlier stages in a sequence of events to modify or examine previously entered data.

At all times the available choices should be clearly displayed. This, of course, calls for some compromise between cluttering the screen by displaying too much information and giving insufficient detail, leaving some doubt as to the effect of each option.

5.1.3. Screen Layout

The layout of the information to be displayed on the screen is a very important part of the design of an interactive graphics program. Several types of information are conveyed to a program user via the display screen, ranging from instructions to graphs and pictures. It is desirable to divide the screen into several distinct logical areas in some standard format which can be maintained as much as possible throughout a program. This increases the speed with which an operator becomes familiar with the information layout and hence makes the program easier and faster to use. A standard screen layout might contain the following areas (Figure 5.1).

1. A major work area used for displaying graphs, numerical results, problem definition, etc.

2. A program identification area. This will usually be situated at the top of the screen and will contain information describing the program in current use for reference purposes.

3. Status information area indicating such things as axes limits, scales and parameter definitions. This area may be included in the work area.

4. An area containing the list of options currently available.

5. A warning and error report area. This may not be separate from the work area but could simply involve flashing the error message over the centre of the screen.

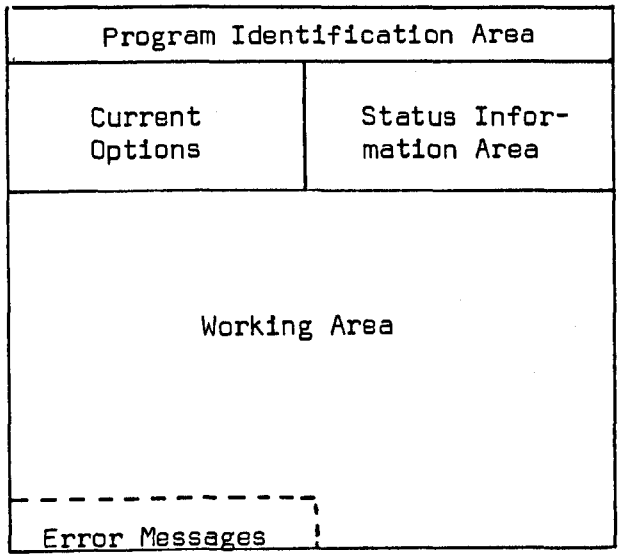


Figure 5.1

It may be that at some stages in a program these areas are not all required. For example, when the method of solution to a problem is being selected the available alternatives can be listed in the centre of the screen using a character size larger than might otherwise have been possible.

As well as standardising screen layout, other simple rules to improve communication can be followed. For example:

(a) Information density should be kept to a realistic level.

(b) Dashed or dotted lines can often be used to clarify a picture.

(c) Small character sizes should be used if they would otherwise overlap or obscure other information.

(d) If the light pen is to be used to select from a list, the items in the list should be well spaced to avoid incorrect selection. Alternatively a user can be asked to verify the selection made before processing continues.

(e) Important information which might change without an operator action (such as error messages) should be positioned in such a way that they will not be obscured by the operator arm.

5.2. Overall Design of an Interactive Graphics System for Solving Differential Equations

Some general principles of design for an interactive graphics system to solve mathematical problems were discussed in the first part of this chapter. These principles will be applied to the area of differential equations and consideration will be given to other more problem-dependent facilities that can be of practical use.

5.2.1. Program Parameters

Firstly, a closer examination will be given to program parameters which will be defined to mean data and user functions (see Chapter 4.5). This excludes on-line decision making, such as the selection of methods of solution, which will be considered as a run time option rather than data. Broadly speaking, anything that could be prepared before run time and supplied on cards will be considered in this category even if it is in fact supplied interactively. For ordinary differential equations (equation 4.1), one-dimensional partial differential equations (equation 4.3) and two-

dimensional partial differential equations defined on a rectangle (equation 4.4), program parameters are used to define the equations, region of solution, initial and boundary conditions, and computational parameters such as step size and error control. In the case of two-dimensional elliptic partial differential equations defined on an arbitrarily shaped region (equation 4.4), and solved using a finite element method only the region of solution and computational parameters can be provided as data. Since the permitted equations have a very simple form they can most easily be defined on-line. It is also easier to define the boundary conditions on-line when the positions and numbers of boundary nodes can be displayed.

For each of the problems above (defined by equations 4.1 to 4.4) program parameters can be either prepared before a run and supplied at load time or they can be supplied completely interactively during a program run. The parameters are defined by a combination of data records and functions. The data records are supplied in a fixed order either on data cards or by typing in the records when they are requested. Functions (used for example to define the equations) can be supplied in one of three ways: using a default function with constants provided as data, using user routines or by typing them in when requested during a run. The input of functions is explained in more detail in Chapter 4.5.

5.2.2. Changing Program Parameters

Since the data is supplied in a standard format it is easily corrected or updated by changing any one (or more) records. This option is made available at various points in the programs where it is logically feasible to change data. The record to be replaced is selected and a new one typed in via the keyboard. Functions can also be changed in a similar way.

5.2.3. On-line Options

Various information is supplied to the programs by a simple selection procedure carried out interactively. For example, to select a method of solution the available methods are listed on the screen each with an associated sense key number. Displayed instructions request the operator to make a choice by depressing the appropriate key. Elsewhere similar techniques are used to select the amount of output to be sent to the lineprinter, the method of input for program parameters and the actual route to be taken through a program. The route options are divided into three types. These are the 'BEGIN' options, available before computation has started, the 'RUN' options, available when intermediate solutions are being computed, and the 'END' options, available when the final solution has been obtained.

The 'BEGIN' options allow data, equations or the method of solution to be changed, hard copies to be taken, or control to be transferred to another point in the program.

The 'RUN' options include choices to continue computation, to change step lengths, to rescale graphs, rotate surfaces or, again, to transfer control.

The 'END' options enable a user to move to the next data set, to continue working with the current data, changing certain parameters if necessary, or to stop.

These options will be described more fully, for each of the programs, in Chapter 6.

5.2.4. Selection of the Variables Displayed

This choice is only available when solving ordinary differential equations (equation 4.1). The user may be interested in the values of the

independent variable, the dependent variables or the errors. A maximum of four of these variables can be selected for numerical display, although if less than four are required they can then be displayed using more significant figures. These values are displayed for the ten most recent steps in the computation.

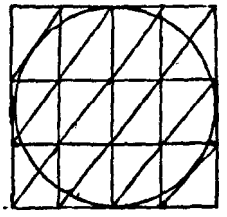
Similarly, up to four graphs may be displayed simultaneously. The variables for each can be selected by the user, and obviously if less than four graphs are displayed (in fact only if the number is reduced to two or one) they can be bigger and hence more detailed.

5.2.5. Light-pen Applications

The light-pen can be used to define the region shape when the finite element method of solution is being used. In the program to be described in Chapter 6.3 the boundary must form a single closed curve and can be 'drawn' with the light-pen. Obviously, this method will not be as accurate as defining the boundary by a set of data points, but it may be much easier to implement if the boundary shape is complicated. The method simply involves moving a tracking cross around the screen with the light-pen and using the sense keys to indicate whether each point selected is to be used, within the program, to fit a curve, a straight line, or to define the beginning or end of the boundary. Fuller details of how to draw a region shape, which may contain curves, corners, and straight lines will be given later.

The light pen is also used when triangles are being fitted to the region of solution before applying the finite element method. The region of solution is firstly triangulated automatically by fitting triangles in a regular way to the smallest rectangle to enclose the region.

For example, for a circular region the display will look like this.



Instructions are then displayed on the screen for fitting the triangles exactly to the region of solution.

Each triangle is numbered so that those completely external to the region of solution can be removed, following the displayed instructions, by typing their numbers on the keyboard at a prescribed point in the program. Those elements remaining with external nodes can then be 'distorted', with the aid of the light-pen so that they lie completely within the region of solution. To locate or identify a particular external node the user simply points at the node with the light-pen and presses a sense key (as instructed on the display screen). The pen is then pointed at a position on the boundary, to which the node is to be 'moved', and again an appropriate key is pressed. The new element or elements are then redrawn automatically before the user can move to the next external node or to the next stage in the program. The criteria for selecting the new nodal position are discussed in Chapter 6.3.

5.2.6. Storage

Data files on disc are used entirely automatically to provide additional storage when insufficient on-line core store is available. For example, a 2-dimensional matrix is used, during the finite element method

of solution, size (n x n) where n is the number of nodes. So, if there are 80 nodes, 12,800 words of storage may be required. This matrix is, therefore, stored in a disc file and each row is brought into the core store as required.

The text used when instructions and options are displayed on the screen is also held on backing store. The block of text used at each stage in a program can then be copied into an array (which must be big enough to hold the largest text block) when it is required. The same array can then be overwritten when a different text block is required.

6. AN INTERACTIVE GRAPHICS SYSTEM FOR DIFFERENTIAL EQUATIONS

In the previous chapters some basic facilities required in an interactive graphics system for solving mathematical problems were discussed with particular emphasis on the solution of differential equations. The three programs, described in this chapter, forming a graphics system for differential equations are GIDODE, for solving ordinary differential equations as defined by equation 4.1, FESOLV for solving elliptic partial differential equations, as defined by 4.2 and GIDPDE for elliptic and parabolic partial differential equations (4.3 and 4.4). The programs are written in FORTRAN for use on the ICL 4130 computer.

Although the structures of the three programs have many common features, each will be described separately in order to preserve the concept of an entire system to solve the three types of problems. However, an overview of the properties common to each program will first be given.

6.1. An Overview of the System

One of the first decisions to be made when planning the design of an interactive graphics program lies in the choice of programming language. Some of the languages used in various applications are discussed in Chapter 2. The programs described later in this chapter are written in FORTRAN despite its known drawbacks, some of which have already been pointed out in Chapter 2.2. There are no ways of overcoming these language difficulties other than attempting to minimise their effect by efficient and strategic programming. Other disadvantages of FORTRAN include a lack of dynamic storage allocation and clumsy facilities for branching, a procedure used extensively in an interactive environment. An attempt is made to overcome

the first of these by holding certain data, not immediately required, on backing store, and by segmenting the program so that only those subroutines in use at any one time are held in core. The modular program structure, with subroutines and segmentation, also helps when branching is necessary although there is no way of overcoming the profusion of 'IF' and 'GO TO' statements present in the programs.

To offset these disadvantages FORTRAN offers certain advantages including the following:

1. It is possible to incorporate routines currently available. For example, the NAG library[44], written for use in FORTRAN and ALGOL 60 programs only, contains validated routines for solving ordinary differential equations.

2. The graphics package DISPAC[18] is written in FORTRAN and NEAT for use only from a FORTRAN program. The details of the package, which resembles the more widely known package GINO-F, are given in Appendix A.

3. FORTRAN is available on all large computer installations so that transfer to a different computer could easily be implemented. It is necessary only to replace the DISPAC routines with similar ones, say GINO-F routines, available on the new machine.

4. FORTRAN is widely known and used so that existing subroutines can be easily replaced by others using new techniques.

Another important factor to consider in the overall design is the role to be played by the user. It is necessary to have a clear idea of what the user expects of the system and what the system demands of the user. In some interactive graphics systems different levels of explanation are provided at each decision point. This means that the inexperienced user who has difficulty understanding brief and often contracted forms used to save space

on a display screen can call for expanded explanations. This kind of facility, however, is very costly in both programming and storage requirements and is felt to be largely unnecessary in the more sophisticated applications considered here. Since it is assumed that the user is familiar with the problem solving process and with the particular problem being solved, words, messages and instructions are chosen to conform as closely as possible with the terminology of differential equations.

If, as suggested above, the user is familiar with the terminology of differential equations, and in addition, has some experience of running computer programs, then no difficulties should be encountered when using these programs. It is perhaps worth mentioning here that the newcomer to computing or differential equations (or both) is not necessarily excluded from using the system, since with very little work the programs could be modified to be of considerable value as teaching aids.

6.2. Hardware Implementation

The hardware available for interactive computer graphics on the ICL 4130 installation at Keele University has been outlined in the first part of Chapter 3. Control of, and interaction with a program using the three hardware communication devices, the sense keys, light pen and keyboard are described in Chapter 3 and Chapter 5.

6.3. GIDODE - A Program to Solve Ordinary Differential Equations

This section describes a program written in FORTRAN for use on an ICL 4180 display terminal. It can be used to solve a system of ordinary differential equations as defined in Chapter 4.1.

The program is divided into three levels each with a set of control options. These allow the user to either operate at the current level or to

jump to another. The program as a whole will be outlined before a detailed account is given of each level.

Level one is the program initialising phase. Before any computation can be carried out the first step is to input program parameters, both data and user functions if required, as well as the computational parameters. The user must also make certain selections from displayed lists using the sense keys. These include deciding how much output is to be sent to the lineprinter and which method of solution is to be used. When initialisation is complete level 1 control options, the 'begin' options are displayed together with a summary of the program and computational parameters already supplied. The user is then able to either correct any errors in the parameters or proceed to another level. It is also possible at this stage to make a permanent copy of the display file containing the problem summary by initiating a copy to the digital plotter.

Level 2 begins with the selection, by the user, of the variables to be displayed in both numerical and graphical form on the screen. This is followed by the computation stage, which takes a different form for the two types of problems, initial or boundary-value problems. In the boundary-value case the system of equations is solved for the whole range but the solution is output on the screen and lineprinter step-by-step. For initial value problems the solution for one step is evaluated and displayed before the level 2 options, the 'run' options are available. These provide facilities for changing computational parameters (for initial-value problems only), initialising a hard copy of the display file, rescaling graphs, changing displayed variables, or simply continuing solution or output for another step. Control can also be transferred to level 3.

Level 3 is the final stage and is reached when a program run for a particular data set is complete. A set of control options, the 'end' options are displayed. The user can then choose to 'rerun' for a perturbed problem by changing a data record. The solution or solutions already obtained can be saved for later display as a comparison. The program can also be run again for a completely new data set, or it can be terminated. Before a detailed description of the three run-time levels the next section will describe the data preparation, giving the layout of data records and the exact forms of the possible user functions.

6.3.1. Data Preparation

The data records can be prepared before running the program and supplied with the control cards when it is loaded. Alternatively, they can be typed in under displayed prompts on the keyboard at run-time. The records take essentially the same form whichever mode of input is used.

An initial-value problem must take the form

$$\frac{dy_i}{dx} = f_i\,(x, y_1, y_2 \cdots y_N)\,, \quad i = 1, 2 \ldots N$$

where the f_is are known functions of the independent variable x and the dependent variables y_i for i = 1, 2 ... N, and where the initial values of these variables, x_0, $y_{1,0}$, $y_{2,0}$ ... $y_{N,0}$ are known. The data records are as follows:

Record Number	Variables	Format
1	XO,H	2FO.O
2	N,EPS	IO,FO.O
3	Y1(O)	FO.O
4	Y2(O)	FO.O
⋮		
N + 2	Y<N>(O)	FO.O

where XO is the initial value of x, H controls the step length along x, N is the number of equations to be solved and EPS is an error control. The variables Y<i>(0) define the initial values of y_i (i = 1, 2 ... N).

EPS is used by the program in the following way. If e_i is the estimated error in y_i then $\frac{ei}{max(y_i)}$ must be less than EPS. Max(y_i) means the maximum value of a particular y_i. This, therefore, gives a relative error control if y_i is growing and an absolute control if it is decaying. Initially max(y_i) is set to unity. The value H is used to specify the maximum step length that will be taken, the minimum will be H x 10^{-2}.

If the system to be solved is a boundary value problem the equations take the same form as above but with the dependent variables specified at each end of a range x_0 to x_N. The equations are then solved for $x_0 \leq x \leq x_N$. The data records are:

Record Number	Variables	Format
1	XMIN,XMAX,R,H,M	4F0.0,I0
2	N	I0
3	A1(1),B1(1),A1(2),B1(2),E1	5F0.0
4	A2(1),B2(1),A2(2),B2(2),E2	5F0.0
⋮		
N + 2	AN(1),BN(1),AN(2),BN(2),EN	5F0.0

where XMIN and XMAX are the initial and final points of the range, R is the matching point, H is the estimated step length and M is the number of points along the range at which the solution is to be output. N is the number of equations. A1(1),A2(1) ... AN(1) contain the known or estimated values of $y_1, y_2 \cdots y_N$ respectively at XMIN, and A1(2),A2(2) ... AN(2) the known or estimated values of $y_1, y_2 \cdots y_N$ at XMAX. B1, B2 ... BN should contain 0.0

if the equivalent element in A<i> is known and 1.0 if it is estimated. E1, E2 ... EN are error bounds for $y_1, y_2, \ldots y_N$ respectively. Tests for convergence use a mixed form so that if tr_i is the estimated modulus of the truncation error in y_i then tr_i < E<i> x (1 + $|y_i|$).

As indicated earlier two methods are provided for defining the equations to be solved. These are, via a user routine or by typing them on the keyboard when requested. A user routine must have a particular name depending on the type of problem being considered. For initial-value problems it must be called DERG and for boundary-value problems DERMN. Each has the form:

```
SUBROUTINE <name> (F,Y,X)
DIMENSION F(<N>), Y(<N>)
F(1)     =  <f1>
F(2)     =  <f2>
 .
 .
 .
F(<N>)   =  <fN>
RETURN
END
```

where N is the number of equations. The f_i's are as defined above, X contains a particular value of x, and Y(1),Y(2) ... Y(N) represent the variables $y_1, y_2 \ldots y_N$ respectively.

If, however, the equations are to be typed in at the prescribed point in the program it is useful to organise them in the necessary form before loading the program. Each must take the form:

$$Y = f_i(X1, X2 \ldots XM)$$

where $M = 2 \times N$

and $X1 \equiv x$, $X2 \equiv y_1$, $X3 \equiv y_2 \ldots X\langle N+1\rangle \equiv y_N$

$X\langle N+2\rangle \equiv y_1'$, $X\langle N+3\rangle \equiv y_2' \ldots XM \equiv y_{N-1}'$

y_i' represents the term $\frac{dy_i}{dx}$ where i = 1, 2, ... N-1

6.3.2. The Initialisation Level

Level 1, the initialisation level provides the facilities, at particular points, for defining the program parameters (data and equations) as well as requiring the user to make on-line selection of the method of solution and amount of lineprinter output. The information provided at this level, (or read from data cards) can be validated by the user before proceeding to level 2.

Once the program GIDODE has been loaded using the control cards described in Chapter 3.3, together with any user functions and data cards, if used, the following information appears in the working area in the centre of the display screen:

```
SELECT TYPE OF O.D.E. SYSTEM AND
       METHOD OF SOLUTION
KEY 1) ADAMS' METHOD - 'NON-STIFF'
          EQUATIONS
    2) MERSON'S METHOD + NEWTON ITERATION -
          TWO-POINT BOUNDARY VALUE PROBLEM
    3) GEAR'S METHOD - 'STIFF' EQUATIONS
```

When the user has selected the required method by pressing the appropriate sense key the information displayed in the working area is replaced by

```
SELECT AMOUNT OF LINEPRINTER OUTPUT
KEY 1) NONE
    2) PART -- VARIABLES DEFINING THE
          PROBLEM AND FINAL SOLUTION
    3) COMPLETE -- AS 2) + SOLN. EVALUATED
          AT EACH 'STEP'
```

When the user has pressed the sense key associated with the quantity of lineprinter output required, the information displayed is replaced by:

```
SELECT INPUT DEVICE FOR EQUATIONS
KEY 1)  TELETYPE
    2)  CARD READER (AS USER FUNCTION)
```

where selecting key 1 indicates that the equations will be typed on the teletype when requested and key 2 indicates that the equations have already been supplied in the form of a user routine. When the selection is complete these instructions are replaced by:

```
SELECT INPUT DEVICE FOR DATA
KEY 1)  TELETYPE
    2)  CARD READER
```

Once this selection has been made a program identification area is set up at the top of the screen containing one of the following titles, as appropriate:

SOLUTION OF INITIAL-VALUE O.D.E. USING ADAMS' METHOD

or SOLUTION OF BOUNDARY-VALUE O.D.E. BY MERSON-NEWTON METHOD

or SOLUTION OF 'STIFF' INITIAL-VALUE O.D.E. USING GEAR'S METHOD

If the teletype is to be used as the data input device the instructions to type the data records replace the current information in the working area of the screen. A list of the variable names and formats for each record to be supplied is also displayed in the working area. For initial-value problems all the data records are input at this stage, but for boundary-value problems only the first two are input here.

If the equations are to be supplied via the teletype the following instruction replaces those displayed in the working area:

```
TYPE EQUATIONS ON TELETYPE
```

Simultaneously Y= is output on the teletype and the first equation can be typed. The request Y= is output before each equation which must be expressed in the form described in Chapter 4.5. The variable names are defined in Chapter 6.3.1. The expressions typed in this way can include the operators + - / * ** (and) which have their usual meanings (** for exponentiation) and the standard functions AINT, ALOG, ALOG10, ATAN, COS, EXP, SIN, SQRT and TANH. Each expression must be terminated with a semi-colon and if any unacceptable expression is typed an error message is output on the teletype. The expression must then be retyped when Y= is output. To delete an expression the user can type % and must then wait for Y= before restarting. The recognisable errors in an expression are as follows:

(a) Illegal function name or variable identifier. This occurs when a variable name is not of the form X<i> or a function name is unrecognisable.

(b) Illegal operator. A character typed is not one of A-Z, 0-9, / + - * ( ) ; or %.

(c) Syntax error. This occurs when

(i) a function name is not followed by (

(ii) two operands are not separated by an operator

or (iii) two operators are not separated by an operand.

The final stage of the initialisation level is the validation stage. A summary of the data together with the 'begin' options replace all the information in the working area (Figure 6.1). These options are self-explanatory. To change a data record or the equations the user must press key 2 after which the list of data records and the following instructions are displayed in the working area:

TYPE 100 TO CHANGE EQNS

or TYPE DATA RECORD NUMBER THEN

TYPE DATA WHEN DEMAND LIGHT IS ON

SOLUTION OF INITIAL-VALUE O.D.E. USING ADAMS' METHOD

TO SOLVE 3 SIMULTANEOUS EQUATIONS

WHERE X(1) = 0 HMAX= 1.-1

Y1 (1) = 1.0

Y2 (1) = 0

Y3 (1) = 0

EPS= 1.-2

BEGIN OPTIONS

KEY 1) CONTINUE

2) CHANGE DATA RECORD OR EQ'NS

3) MOVE TO END OPTIONS

4) PLOT TO DIGITAL PLTER

8) STOP

Figure 6.1.

Normally only one data record can be changed each time key 2 is selected. The only exception arises when N, the number of equations to be solved (defined on record 2) is changed. In this case it is necessary to retype all the following records whether or not they are to be changed.

When the instructions to change the equations or data have been fulfilled control returns to the 'begin' options, which are again displayed with a summary of the data. If the user is satisfied with this data, processing is passed to level 2 by pressing key 1. Level 2 is principally the computation level but the user must begin by selecting the variables to be displayed numerically and graphically.

6.3.3. Level Two

Instructions for the selection of variables for numerical output are as follows:

```
        SELECT VARIABLES FOR DISPLAY
(1)     NUMERICAL DISPLAY (MAX OF 4)
        KEY 1)  TO SELECT VARIABLES -- TYPE NOS.
                FORMAT (IO) END WITH 0
           6)   FOR UNCHANGED CHOICE
           7)   DEFAULT X, Y1 AND E1
```

Each variable X, Y1, Y2 ... Y<N>, E1, E2, ... E<N> is displayed, down the right side of the screen, with an associated number. If the default variables are insufficient an alternative selection is made by pressing key 1 and then typing the associated numbers of the chosen variables as integers on the teletype. Each number must be typed on a separate line, the final line containing the terminating value zero. Key 6 can be used if a problem or a perturbed problem is being rerun and the display variables are to remain unchanged. If only one variable is to be displayed numerically

a field width of eight digits is used. This is reduced by one for each additional variable. When the selection of variables for numerical display is complete the numbered variable list remains on the screen but the previous instructions are replaced by:

```
     SELECT VARIABLES FOR DISPLAY
(2)  GRAPHICAL DISPLAY (MAX OF 4)
     KEY 1)  TYPE NO. OF GRAPHS (IO) THEN
             VARIABLE NUMBERS FOR EACH (2IO)
             (HORIZONTAL AXIS FIRST)
        6)  FOR UNCHANGED CHOICE
        7)  DEFAULT -- 2 GRAPHS X-Y1
                AND X-E1
```

To select variables, other than the defaults, key 1 must be pressed and the total number of graphs required typed, as an integer, on the teletype. This must be followed by pairs of variable-associated numbers, defining the axes of each graph, where each pair of numbers is typed on separate lines as integers separated by a comma. The area of the screen used when only one graph is displayed is approximately six inches square, for two graphs the area for each is half this size, and for three or four graphs the area is scaled by a quarter.

The variables selected for numerical or graphical display can be changed dynamically later in the program if required.

Unless the graphical display variables are unchanged it is necessary to define lower and upper limits for each variable being plotted. Instructions in the form:

```
TYPE BOUNDS FOR X AXIS
```

are displayed for each variable used. These bounds can be changed at any time during a run by rescaling.

For a boundary-value problem it is necessary at this stage, if the data is being typed in, to supply the remaining records defining the boundary conditions. The variables and formats are displayed as described above and when the data has been typed a summary is shown for checking and if necessary for correction.

At this point in the program, computation processing actually begins. For a boundary-value problem the system of equations is automatically integrated over the whole range, and when integration is complete the solution is displayed, for the first output point, in the working area. For initial value problems the solution is evaluated for only one step along x and output for that point.

The options available between each step are

RUN OPTIONS

KEY 1) CONTINUE

2) CHANGE STEP LENGTH

3) RESCALE

4) PLOT TO DIGTL PLTER

5) MOVE TO END OPTIONS

6) CHANGE DISPLAY VARIABLES

7) CHANGE EPS

Also displayed between each step is the actual step size used (H) and for initial-value problems only, the maximum step size (HMAX), the error control (EPS) and the number of steps taken (STEPS). Figure 6.2 shows a typical display file at this stage. The dotted lines on each graph are 'saved' solutions of a 'similar' problem.

Some of the alternatives available under the 'run' options will be explained briefly. The integration process (or output in the case of a

SOLUTION OF 'STIFF' INITIAL-VALUE O.D.E USING GEAR'S METHOD

RUN OPTIONS

KEY 1) CONTINUE
2) CHANGE STEP LENGTH
3) RESCALE
4) PLOT TO DIGTL PLTER
5) MOVE TO END OPTIONS
6) CHANGE DISPLAY VARIABLES
7) CHANGE EPS

H=7.186.0
HMAX=1.1
EPS=1.-3
STEPS= 35

X	Y1	E1
1.2.1	8.-1	5.-5
1.5.1	8.-1	5.-4
1.7.1	8.-1	9.-4
2.1	8.-1	8.-4
2.3.1	8.-1	5.-4
2.7.1	8.-1	7.-4
3.1.1	7.-1	6.-4
3.5.1	7.-1	4.-4
3.9.1	7.-1	3.-4
4.6.1	7.-1	6.-4

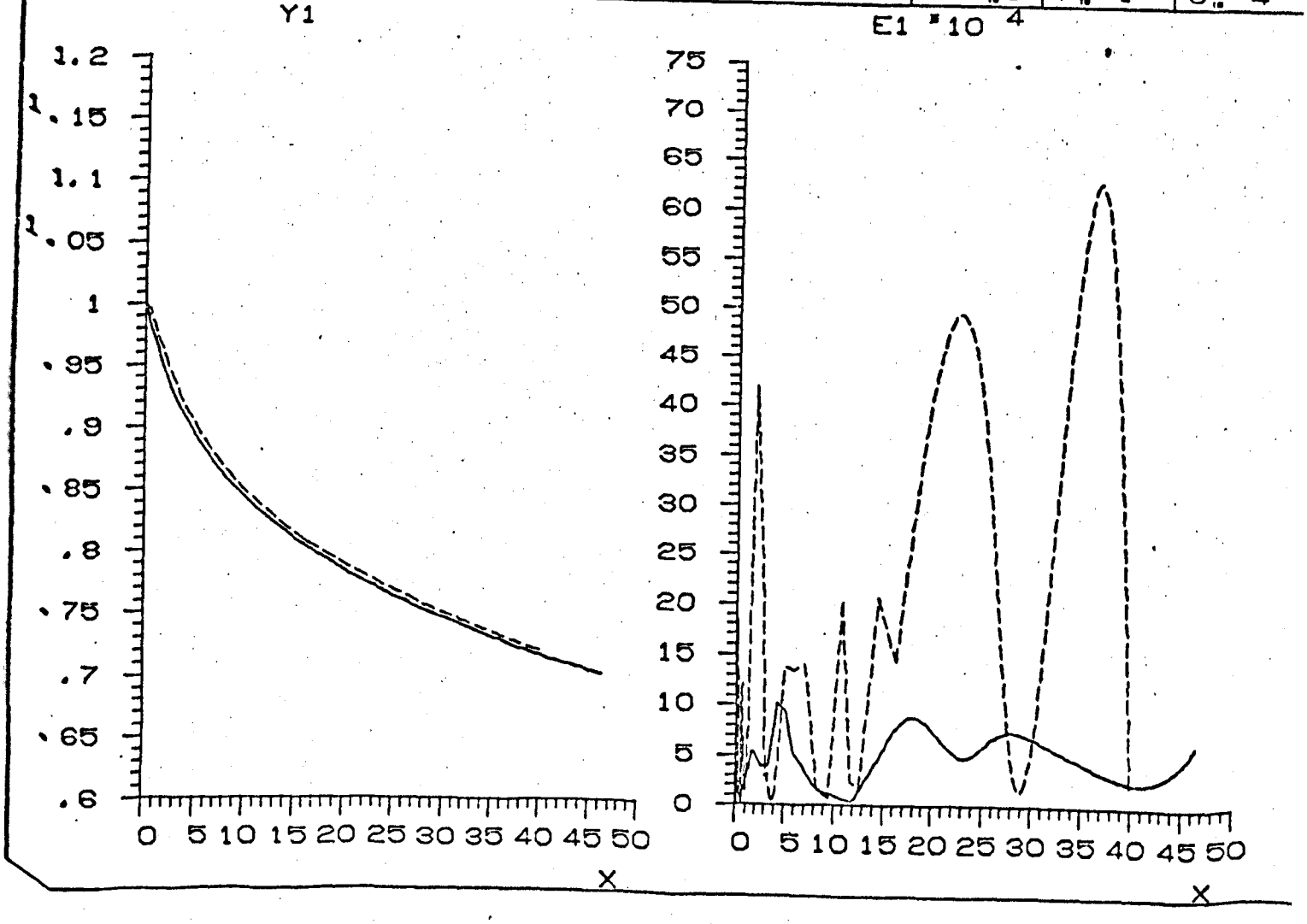


Figure 6.2.

boundary-value problem) continues automatically for one step when key 1 is pressed. The solution after this step is added to the numerical display and to the arrays used to generate the displayed curves. Up to 81 steps can be evaluated and displayed graphically. To continue for more than one step, key 1 is raised and left in the 'on' position. (See Chapter 3.1). It must, however, be switched 'off' (horizontal position) before another key can be read.

Keys 2 and 7 are available only for initial-value problems. Key 2 is used to change the maximum step length along x by typing in a new value. Similarly, EPS can be changed if key 7 is selected. Key 3 is used to rescale one of the axes displayed. When activated the following instructions are displayed:

TYPE THE NUMBER OF THE VARIABLE
TO BE RESCALED

THEN TYPE MIN AND MAX VALUES

accompanied by a list of variable names and associated numbers. Key 6 is used to change the variables displayed. When pressed the instructions for selecting the display variables as previously described are displayed in the working area. After any choice except key 5 control remains with the 'run' options. If at any time a failure occurs in one of the routines solving the equations, an appropriate error message such as 'EPS TOO SMALL' will be displayed and the necessary changes can then be made.

6.3.4. Level Three - The Finalisation Level

When the solution of a particular problem is complete, key 5 is used to transfer control to level 3 where the 'end' options are available. These

are displayed on the screen in the working area (replacing the 'run' options, and numerical and graphical output) as follows:

END OPTIONS

KEY 1) MOVE TO NEXT DATA SET

2) CHANGE DATA, EQNS. OR METHOD: RERUN

3) AS 2) + RETAIN CURRENT SOLN.

4) AS 3) + RETAIN ALL PREVIOUSLY RETAINED SOLNS.

8) STOP

If key 2), 3) or 4) is used the 'end' options are replaced by the following instructions:

TYPE 20 TO RERUN

TYPE 100 TO CHANGE EQNS

TYPE 200 TO CHANGE METHOD

or TYPE DATA RECORD NUMBER THEN

TYPE DATA WHEN DEMAND LIGHT IS ON

Appropriate instructions are then displayed depending on which parameter (data, equations or method of solution) is to be changed. Once the particular change has been made control returns to level 1 and the initiating 'begin' options.

Applications of the various alternatives available in GIDODE will be described in the next chapter. In Chapter 8 some suggestions for improving the system, in the light of experience gained in these and other applications, will be discussed.

6.4. GIDPDE - A Program to Solve Partial Differential Equations

This section describes the interactive graphics program GIDPDE written in FORTRAN to solve the partial differential equations defined by equations 4.3 and 4.4 in Chapter 4. The equations are solved using the finite difference methods of Crank-Nicolson and Peaceman-Rachford which are described in Appendix B. This program, like GIDODE is divided into three levels each containing a set of control options. The levels are the initialisation level, computation level and finalisation level.

Level 1, as in GIDODE, is the initialising level. At this level the program parameters (data and user functions) are input as well as the computation parameters. The data and functions necessary to define a problem can be either provided on cards or typed on the keyboard, at prescribed points in the program. The user must also make on-line selections, from lists of alternatives displayed on the screen, of the amount of output required on the lineprinter and the type of equation to be solved. When the initialisation of parameters is complete the set of 'begin' options, similar to those available in GIDODE, is displayed on the screen together with a summary of the data. The data can then be validated before the user proceeds to level 2, the computation level.

At level 2, the computation of the solution begins. A set of 'run' options similar to those described for ordinary differential equations are available between each solution step. They enable the user to change certain parameters or to transfer control to level 3, the finalisation level.

Level 3 is again similar to level 3 in GIDODE and contains the 'end' options.

Before a detailed account of each level the necessary form of the data and user functions will be described.

6.4.1. Data Preparation

This includes the definition of equations, initial and boundary conditions, other parameters necessary to define the problem, and computation parameters.

(A) Equation Definition

The coefficients of the equation to be solved must be supplied by the user at a prescribed point when the program is running. The various forms possible for defining these coefficients will be described later.

(B) Initial Conditions

Initial conditions can be defined using the default function or by supplying a user function. The default functions for the initial values of the dependent variable u_o are

(a) One-dimensional problem

$$u_o = f_1 + f_2 x$$

(b) Two-dimensional problem

$$u_o = f_1 + f_2 x + f_3 y$$

The values of the constants f_1, f_2 and f_3 must be supplied by the user with the data.

The user functions for the initial conditions must be of the form

```
FUNCTION UINIT (L,NF,X,F)
```

where L - is not used here but must be included so that the user function is of the same form as the default function.

NF - is the length of the array F and must be ≥ 1.

X - (i) 1-D problem. X is a real variable containing a particular value of x.

X - (ii) 2-D problem. X is an array length 2, where X(1) ≡ x and X(2) ≡ y.

F - a real array, length NF, containing certain constants input as data. F need not be used (see Chapter 4.4).

(C) Boundary Conditions

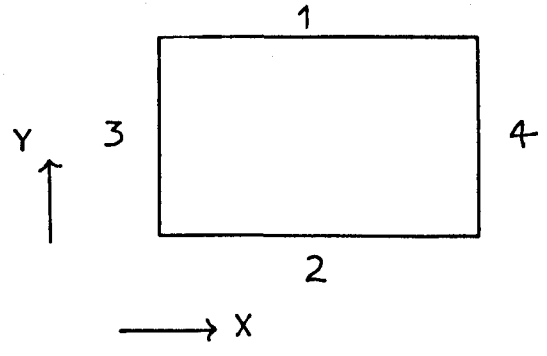
When time-dependent, or, for a two-dimensional problem, position-dependent, boundary conditions are required either the default function or a user function can be used. The default functions for the boundary values u_b are

(a) One-dimensional problem

$$u_b(t) = BN_1 + BN_2 t$$

(b) Two-dimensional problem.

The boundaries must be numbered in the order shown



For boundary 1 and 2 the function is

$$u_b(x_1 t) = BN_1 x + BN_2 t$$

and for 3 and 4

$$u_b(y_1 t) = BN_1 y + BN_2 t$$

The values for BN_1 and BN_2 must be provided on the appropriate data record for each boundary that they are required.

The user functions, for defining the boundary conditions must be in the form

(a) one-dimensional problem

```
FUNCTION BNDT (BN,T)
```

(b) two-dimensional problem

```
FUNCTION BNDZT (BN,X,T)
```

where BN is a real array length 2 contain constants that may be supplied by the user on the appropriate data record. These can of course be specified in the user function but it can sometimes be useful to provide them separately (with the data) so that a user function can be used for more than one boundary.

X ≡ x for boundaries 1 and 2.

≡ y for boundaries 3 and 4.

T ≡ t (time). For elliptic two-dimensional partial differential equations T is redundant.

(D) Data records

Data records can be provided on cards or they can be typed on the keyboard at prescribed points in the program. The variables on each record are independent of the mode of input but the format is fixed for cards input and free for keyboard input. The fixed format simplifies data checking when preparing cards, and free format, for keyboard input, means that instead of typing spaces between numbers they can be separated simply by a comma. The layout of each record will be described for card input, but for keyboard input it is simply necessary to replace the fixed format by a free format. For example, (I2,2F10.0) will be replaced by (I0,2F0.0).

The data records will be described separately for a one-dimensional and a two-dimensional problem.

(i) One-dimensional parabolic partial differential equation.

The boundary conditions are each read in on a card of CARD 1 format. The condition for minimum x is input first.

CARD 1. (I2,2F10.0)

	columns		
I2	1 - 2	NB = - 1	for a time dependent boundary value with constants BN_1 and BN_2 used in the default function or in a user function
		= 1	for a constant boundary value BN_1
		= 2	for an insulated boundary $\frac{\partial u_b}{\partial x} = 0$
		= 3	for a radiative boundary $\frac{\partial u_b}{\partial x} = \pm h(u_b - v)$ where $h = BN_1$ and $v = BN_2$ u_b is the value of u at the boundary
2F10.0	3 - 12	BN_1	constants required to define the boundary condition
	13 - 22	BN_2	

CARD 2. (2F10.0)

	columns		
	1 - 10	XMIN	minimum and maximum value of the independent variable x.
	11 - 20	XMAX	

CARD 3. (I2,2F10.0)

	columns		
I2	1 - 2	N	where N is the number of steps along x. (N < 241).
2F10.0	3 - 12	TSTART	initial value of t (time)
	13 - 22	DT	time step

CARD 4. (2F10.0)

columns		
1 - 10	F1	Real constants used in the default function or the user function UINIT to specify the initial values of u.
11 - 20	F2	

(ii) Two-dimensional elliptic and parabolic partial differential equations.

The constants required to specify the boundary conditions are read in on four cards in CARD 1 format in the order 1, 2, 3, 4 for the boundary numbering system described above.

CARD 1. (I2,2F10.0)

	columns		
I2	1 - 2	NB = - 1	default or user function
		= 1	constant
		= 2	insulated $\frac{\partial u_b}{\partial n} = 0$
		= 3	radiative $\frac{\partial u_b}{\partial n} = \pm h(u_b - v)$
			where $\frac{\partial u_b}{\partial n}$ is the normal derivative at $u = u_b$
2F10.0	3 - 12	BN_1	
	13 - 22	BN_2	

(See CARD 1 for a one-dimensional problem for a fuller description of each variable).

CARD 2. (4F10.0)

columns		
1 - 10	XMIN	
11 - 20	XMAX	where XMIN $\leq x \leq$ XMAX
21 - 30	YMIN	YMIN $\leq y \leq$ YMAX
31 - 40	YMAX	

CARD 3. (2I2,2F10.0)

	columns			
I2	1 - 2	N	Number of steps along x	
	3 - 4	M	Number of steps along y	
			Parabolic/Elliptic	
2F10.0	5 - 14		TSTART / RHO	TSTART is the initial value of t (time) and RHO the acceleration parameter
	15 - 24		DT /	DT is the time step

CARD 4. (3F10.0)

columns		
1 - 10	F1	Real constants used to define the initial
11 - 20	F2	values of u in the default function or
21 - 30	F3	user function UINIT

Equation definition is carried out entirely on-line where the coefficients are input as constants or functions.

6.4.2. Initialisation Level

The program GIDPDE is loaded using the control cards described in Chapter 3.3 together with any data and user functions required.

The following instructions then appear in the centre of the screen. This area will be called the working area.

```
SELECT AMOUNT OF LINEPRINTER OUTPUT
KEY 1)  NONE
    2)  PART -- VARIABLES DEFINING THE PROBLEM
                AND FINAL SOLUTION
    3)  COMPLETE -- AS 2) + SOLN EVALUATED AT
                EACH 'STEP'
```

When the user has pressed the key appropriate to his requirement these instructions are replaced by

SELECT TYPE OF P.D.E.

KEY 1) PARABOLIC 1-DIMENSION

2) PARABOLIC 2-D RECTANGULAR DOMAIN

3) ELLIPTIC 2-D RECTANGULAR DOMAIN

When the type of equation has been selected the next stage of the initialisation level, the definition of the equation coefficients, is reached. The coefficients are A_1, A_2, A_3 and A_4 for equation 4.3 and A_1, A_2 ... A_6 for equation 4.4.

The following instructions replace those displayed on the screen in the working area:

INPUT EQN. COEFFICIENT A<i>

KEY 1) DEFAULT VALUE 0.0

2) DEFAULT VALUE 1.0

3) CONST. OR POSN. DEPENDENT (TYPE IN)

4) TIME DEPENDENT (TYPE IN)

When the coefficient A<i> is zero or unity the user can select and press key 1 or 2 respectively. However, coefficients having other constant values, or functions of x, y or t must be typed in on the keyboard. To do this the user must firstly press key 3 or 4. Key 3 if the function defining a coefficient is not time dependent and key 4 otherwise. The symbols Y = will then appear on the teletype. The function or expression can then be typed. Only the variable names X1, X2 and X3 can be used and these are defined in the following way.

(i) One-dimensional problem

X1 ≡ x; X2 ≡ t

(ii) Two-dimensional problem

X1 ≡ x; X2 ≡ y; X3 ≡ t

SOLUTION OF LINEAR PARABOLIC P.D.E. IN ONE SPACE DIMENSION

TO SOLVE $$\frac{DU}{DT}=A1*\frac{D}{DX}\left(\frac{DU}{DX}\right)+A2*\frac{DU}{DX}+A3*U+A4$$

WHERE A1= 1 A2= 0

A3= 0 A4= 0

BOUNDARY CONDITION AT X= 0

DU/DX=1 * (UB- 0) XMIN= 0 XMAX= 1 H=.25

TMIN= 0

BOUNDARY CONDITION AT X= 1

DU/DX=-1 * (UB- 0)

BEGIN OPTIONS

KEY 1) CONTINUE

2) CHANGE DATA RECORD OR COEFFS

3) MOVE TO END OPTIONS

4) PLOT TO DIGITAL PLTER

8) STOP

Figure 6.3.

The necessary form of an expression and the error messages are the same as those described for GIDODE in Chapter 6, Section 3.2.

When the definition of equation coefficients is complete the instructions in the working area are replaced by

```
SELECT INPUT DEVICE FOR DATA
KEY 1)  TELETYPE
    2)  CARD READER
```

If the data is going to be typed in on the keyboard key 1 must be pressed. These instructions are then replaced by a list of the variables and formats on each data record and the instruction

```
TYPE DATA RECORDS
```

When the input of data records is complete, a summary of the problem definition for validation is displayed, with the 'begin' options, in the working area. A program identification area is set up at the top of the screen in which one of the following titles is displayed:

```
SOLUTION OF LINEAR PARABOLIC P.D.E. IN ONE SPACE DIMENSION
SOLUTION OF LINEAR PARABOLIC P.D.E. IN TWO SPACE DIMENSIONS
SOLUTION OF LINEAR ELLIPTIC P.D.E. IN TWO SPACE DIMENSIONS
```

The display file summarising level 1 operations for a typical problem is shown in Figure 6.3. The 'begin' options are very similar to those described for GIDODE. To correct any errors key 2 must be pressed. The following instructions will then be displayed in the working area, together with the numbered list of data records:

TYPE 100 TO CHANGE COEFFS

or TYPE DATA RECORD NO. (20 TO RERUN)

then TYPE DATA WHEN DEMAND LIGHT IS ON

If '100' is typed on the keyboard the instructions described above for the definition of coefficients are displayed in the working area.

When the redefinition of coefficients or the input of a new data record is complete control returns to the 'begin' options which are displayed with the new data summary. When the user is satisfied with the data, key 1 is used to proceed to the second level, the computation level.

6.4.3. The Computation Level

Before the computation of solution values actually begins, for a one-dimensional problem it may be necessary for the user to supply estimates for the expected minimum and maximum values of u, the dependent variable. This is only necessary when the boundary conditions are such that it is not possible to do so automatically. These values are used to scale the u-axis for display purposes and can be changed by interactively rescaling the axis at some later stage in the computation. If it is necessary to supply these values, the following instructions are displayed in the working area:

TYPE UMIN AND UMAX

Calculations corresponding to one time step or iteration are carried out automatically and the results displayed in numerical and graphical form. After this and each subsequent step the numerical values of t, the time, for a parabolic equation or IT, the iteration count, for an elliptic equation are displayed together with the value of the maximum percentage difference in a grid value of u, the dependent variable. The ten most recent values for these variables are displayed at one time.

Graphical output consists, for a one-dimensional problem, of a graph of u against x with the solutions plotted for the most recent three time steps. For a two-dimensional problem the surface u(x, y) for the current iteration or time value is plotted as a projection onto the plane of the screen.

Between each step the 'run' options are available, these are displayed on the screen with the numerical and graphical output. For a one-dimensional problem these are:

```
          RUN OPTIONS
KEY 1)  CONTINUE
    2)  CHANGE TIME STEP
    3)  RESCALE
    4)  PLOT TO DIGTL PLTER
    5)  MOVE TO END OPTIONS
```

For a two-dimensional parabolic equation they are:

```
          RUN OPTIONS
KEY 1)  CONTINUE
    2)  CHANGE TIME STEP
    3)  RESCALE
    4)  PLOT TO DIGTL PLTER
    5)  MOVE TO END OPTIONS
    6)  ROTATE - TYPE NO. DEGS (IO)
```

and for an elliptic problem option 2 ('Change time step') of those for the two-dimensional parabolic case is replaced by 'CHANGE ACCN PRMTR'. An example of the display file for a one-dimensional parabolic problem is shown in Figure 6.4.

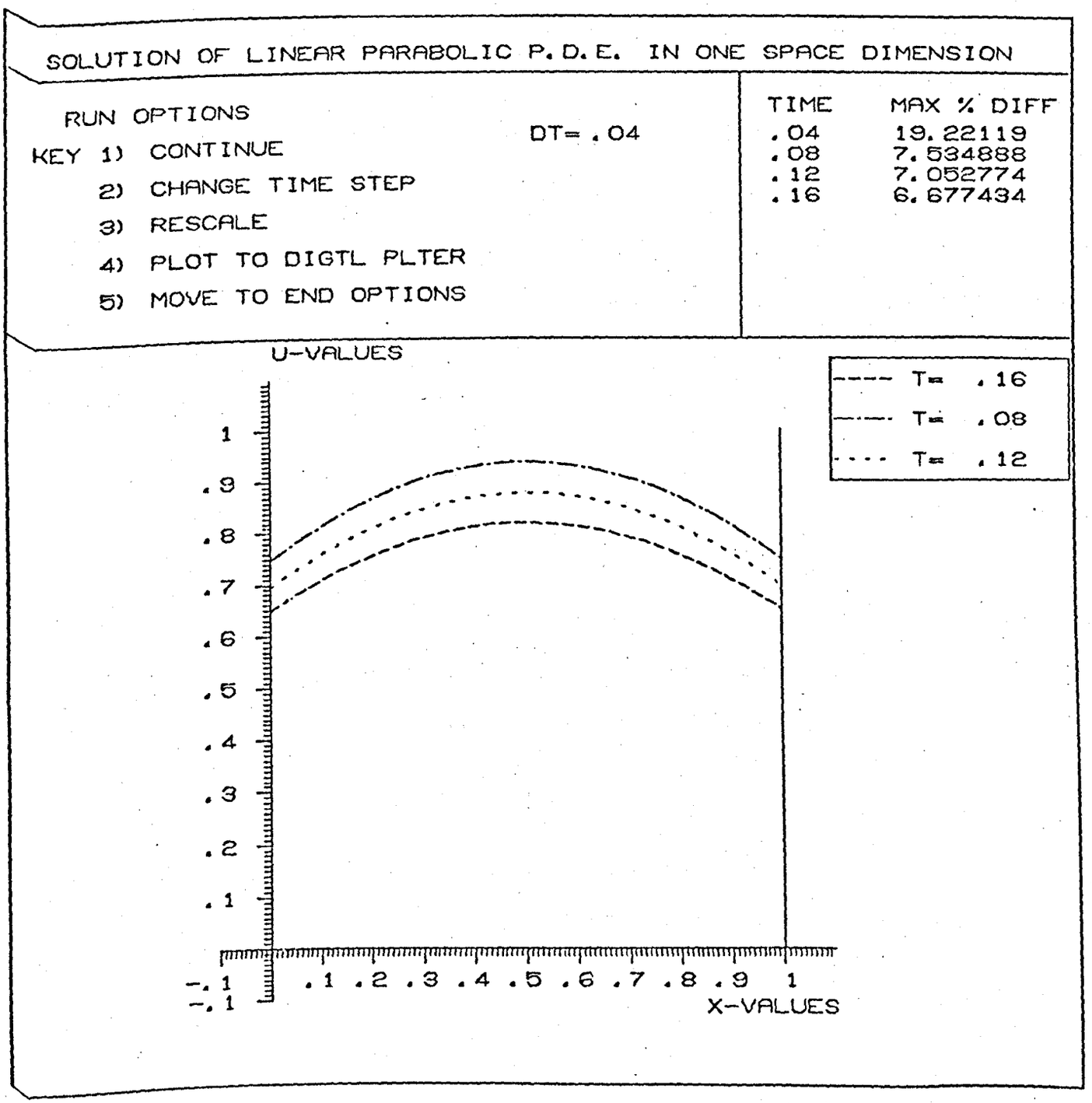

SOLUTION OF LINEAR PARABOLIC P.D.E. IN ONE SPACE DIMENSION
RUN OPTIONS
DT= .04
KEY 1) CONTINUE
2) CHANGE TIME STEP
3) RESCALE
4) PLOT TO DIGTL PLTER
5) MOVE TO END OPTIONS
TIME MAX % DIFF
.04 19.22119
.08 7.534888
.12 7.052774
.16 6.677434
U-VALUES
T= .16
T= .08
T= .12
1
.9
.8
.7
.6
.5
.4
.3
.2
.1
-.1
-.1 .1 .2 .3 .4 .5 .6 .7 .8 .9 1
X-VALUES

Figure 6.4.

During any of these 'run' options key 1 is used to continue the solution for another step or iteration. The key can be raised and, therefore, left in the 'on' position to avoid activating it between each step. It must, however, be switched 'off' before another key can be used. Appropriate instructions are displayed when key 2, 3 or 6 is selected. Rotation of a displayed surface is anti-clockwise in the x-y plane through N degrees, where N is typed on the teletype. To return the surface to its original view N must be given the value -1. When the final time step or iteration is reached, key 5 transfers control to the finalisation level.

6.4.4. Finalisation Level

The third level, the finalisation level, contains the 'end' options. These replace the numerical and graphical output, and the 'run' options previously displayed in the working area of the screen. They are:

END OPTIONS

KEY 1) MOVE TO NEXT DATA SET

2) CHANGE DATA OR COEFFS + CONTINUE

3) CHANGE DATA OR COEFFS + RERUN

4) ESTIMATE ERROR IN FINAL SOLUTION

5) PLOT ERROR CURVE/SURFACE TO D.P.

with, for a one-dimensional problem:

6) AS 2) + RETAIN CURRENT SOLUTION

7) AS 3) + RETAIN CURRENT SOLUTION

8) STOP

or, for a two-dimensional problem:

6) ROTATE - TYPE NO. DEGS (IO)

8) STOP

When key 4 is selected the error terms in the final solution are evaluated and the error curve or surface is plotted on the screen. A permanent copy is obtained using key 5 and, for a two-dimensional problem, the error surface is rotated using key 6. The conditions of the rotation are the same as those described in the 'run' options. After key 4 or 5, or 6 for a two-dimensional problem, control remains with the 'end' options. Key 1 transfers control to the beginning of the initialisation level and a new problem can be started. Keys 2 and 3 (and 6 and 7 for a one-dimensional problem) are followed by the instructions described for key 2 of the 'begin' options which also allows a change of data or coefficients. When the data or coefficients have been changed control is transferred to the 'begin' options in level 1.

A typical route through GIDPDE is described in Chapter 7 to illustrate some of the facilities described here.

6.5. FESOLV - A Program to Solve Elliptic Partial Differential Equations

The program FESOLV can be used to solve certain two-dimensional elliptic partial differential equations over a wide range of region shapes, as defined by equation 4.2 in the first part of Chapter 4. The structure of the program is rather different from the two previous programs GIDODE and GIDPDE. The finite element method, which is described in Appendix B, does not involve steps or iterations so no intermediate solutions are available. The emphasis, from an interactive graphics point of view, is placed on defining the region of solution and the elements, or triangles, into which it is

divided, rather than on interaction and graphical display during the solution stage. More attention is also given to ways of displaying the final solution. This is necessary since it is rather more difficult to clearly display a three-dimensional figure defined within an arbitarily shaped region than one defined on a rectangle.

The program can again be divided into three levels, of initialisation, computation and finalisation. However the computation stage is entirely automatic, interaction being used only to set up the problem and within the finalisation level.

6.5.1. Data Preparation

Information defining the region of solution and the triangle size can be supplied either on cards, using the light pen and sense keys at run time, or from a disc file. If the data is to be supplied on cards, when the program is loaded, the records take the following form:

Record No.	Variables	Format
1	REGION	A8
2	XP_1, YP_1	2F0.0
3	IP	I0

REGION contains up to eight characters used to name the particular problem being solved. This name can be used for identification purposes as the file name if information is to be stored on disc.

XP_1, YP_1 is the starting point on the boundary curve. Since the boundary must be defined by a single closed curve this point is arbitrary.

IP defines the type of boundary shape.

IP = 1 for a region defined by a number of points to be supplied and joined together by straight lines.

= 2 for a single closed curve to be fitted to a number of supplied points using parametric cubic splines.

IP = 3 for a region shape defined by a combination of 1 and 2.

The remaining data records, except the last depend on the value of IP.

(a) IP = 1

Record No.	Variables	Format
4	NPER	I0
5	XP_2, YP_2	2F0.0
⋮	⋮	⋮
NPER+4	XP_{NPER+1}, YP_{NPER+1}	2F0.0

NPER defines the number of points to be supplied (< 46). XP_1,XP_i for i = 2, ... NPER+1 - Points to be jointed together by straight lines. (XP_1,YP_1) is also joined to (XP_2,YP_2) and to (XP_{NPER+1}, YP_{NPER+1}).

(b) IP = 2

Record No.	Variables	Format
4	NP,NPER	2I0
5	SX_2,SY_2	2F0.0
⋮	⋮	⋮
NP+4	SX_{NP+1}, SY_{NP+1}	2F0.0

The NP(< 16) points (SX_i,SY_i) for i = 2, ... NP+1 together with ($SX_1 = XP_1$, $SY_1 = YP_1$) and ($SX_{NP+2} = XP_1$, $SY_{NP+2} = YP_1$) are fitted to a curve using parametric splines. NPER (< 46) equally spaced points on the curve are joined together by straight lines.

(c) IP = 3

Record No.	Variables	Format
4	NP,NPER	2I0

NP defines the type of curve to be used to define the part of the boundary between the last point of the previous curve (or (XP_1,YP_1) for the first curve) and the NP following points.

NP = 0 Boundary complete.

NP = 1 The last point to be joined to the one following point by a straight line. NPER is not required.

NP > 1 (and < 16). The following NP points together with the last point are fitted to a curve with parametric splines NPER (< 46) equally spaced points on the curve are joined together with straight lines.

If NP = 1 the following record is

Record No.	Variables	Format
5	XP_j, XP_j	2F0.0

If NP > 1 the next records are

Record No.	Variables	Format
5	SX_2, SY_2	2F0.0
⋮	⋮	⋮
NP+4	SX_{NP+1}, SY_{NP+1}	2F0.0

The sequence "Record 4 to NP+4" is repeated until NP = 0.

The final record for all values of IP is

Variables	Format
NL,NH	2I0

where NL = number of triangle widths in x-direction

and NH = number of triangle heights in y-direction.

When the region of solution is to be defined on-line the records take a similar form but types of curves are selected using the sense keys and actual points are supplied by moving a tracking cross with the light pen to the desired point. Instructions for defining the region are displayed on the screen.

The data, triangles and solutions can be written onto disc so that they can be read from disc for subsequent runs. Since the solution of a problem using finite elements may take a considerable time this may be useful if a user wishes to rotate, plot or simply examine the solution at a later time.

6.5.2. Initialisation Level

The first choice to be made on-line is again the amount of output to be sent to the lineprinter. The following instructions are displayed on the screen in the working area:

```
SELECT LINEPRINTER OUTPUT
KEY 1)  NONE
    2)  PART - PROBLEM + SOLN AT EACH NODE
    3)  COMPLETE - AS 2) + TRIANGLES
```

When the selection has been made this step is followed by the selection of the mode of data input and the storage requirements. The instructions are replaced by

```
INPUT DEVICE AND DISC STORAGE
KEY 1)  DATA, TRIANGLES + SOLN FROM DISC
    2)  DATA FROM CARDS/DATA ETC. TO DISC
    3)  DATA FROM CARDS/NO STORAGE
    4)  DATA VIA T'TYPE + LIGHT PN/NO ST'AGE
```

Key 1 is used to read the data, triangle vertices and solutions from disc, assuming, of course, that they have previously been written to a disc file. When key 4 is selected instructions for the user to supply a problem name and estimated values for bounds of x and y are displayed before the instructions for defining the region. The instructions for the region

definition are accompanied by a tracking cross for tracing the curve bounding the region with the light pen, and a scaled, labelled set of axes superimposed with a grid. The instructions are as follows:

DEFINE REGION SHAPE
MOVE CROSS TO NEW POINT AND PRESS KEY
KEY 1) STARTING POINT
2) DRAW LINE - LAST PT. TO NEW PT.
3) STORE POINT (IN STR)
4) FIT CURVE TO STORED PTS AND CLEAR STR.
5) END (WHEN LAST PT. = FIRST PT.)

The starting point must be selected first. When a region shape can be defined by a set of straight lines it is necessary to move the tracking cross to the end of each line around the boundary (the first line begins at the 'starting' point) activating key 2 each time. Alternatively, a boundary can be made up of one or more smooth curves, or a combination of smooth curves and straight lines. A curve is generated by storing successive points along it in STR using key 3. When the end of the curve is reached, key 4 is activated and the curve will be drawn automatically using parametric splines. The next curve can then be started or a straight line drawn. When the boundary is complete, that is, when the last point is the same as the starting point, key 5 must be pressed. The following is then displayed:

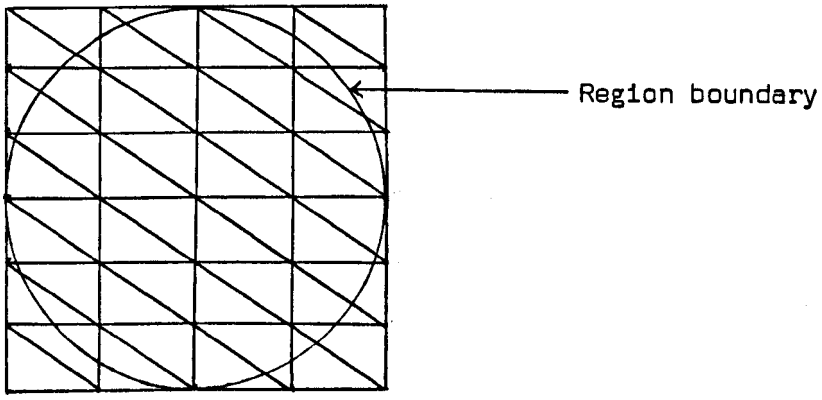
TYPE NL AND NH (NO. T'NGLES ALONG X AND Y)

The integers NL and NH must then be typed on the keyboard (format 2I0).

When the region shape and triangle size have been defined, the next stage is to fit the triangles to the region of solution. The first step in this process is automatic. The region is drawn onto the screen and

triangles are fitted in a regular pattern to the smallest rectangle enclosing the region. Each triangle is numbered.

For example, if NL has been defined to be 4 and NH to be 6, then, given a circular region, the automatic triangulation will be



The following instructions are displayed in an instruction area, above the triangularised region.

```
FIT TRIANGLES TO REGION
KEY 1)  REMOVE EXTERNAL ELEMENTS
    2)  MOVE EXT NODE TO BNDRY (AFTER 1)
    3)  CONTINUE (TO EQN DEFINITION)
    4)  ADD NODE BISECTING AN EDGE
    5)  MOVE TO NEXT DATA SET
    6)  PLOT
    8)  STOP
```

The display file at this stage is shown in Figure 6.5 for an example problem. The next step is to remove all elements that are wholly external to the region of solution. This step is initiated by activating key 1 after which the following instruction is displayed in place of the previous instructions. The triangularised region remains on the screen.

```
TYPE EXTERNAL ELEMENT NOS. (IO) ENDING WITH ZERO
** TO CORRECT ERROR -- TYPE -1 AND RESTART **
```

TO SOLVE $-\frac{D}{DX}\left(\frac{DU}{DX}\right) - \frac{D}{DY}\left(\frac{DU}{DY}\right) + G*U = F$

FIT TRIANGLES TO REGION

KEY 1) REMOVE EXTERNAL ELEMENTS
2) MOVE EXT NODE TO BNDRY (AFTER 1)
3) CONTINUE (TO EQN. DEFINITION)
4) ADD NODE BISECTING AN EDGE
5) MOVE TO NEXT DATA SET
6) PLOT
8) STOP

REGION NAME -

CIRCLER1

NO. OF TRIANGLES

50

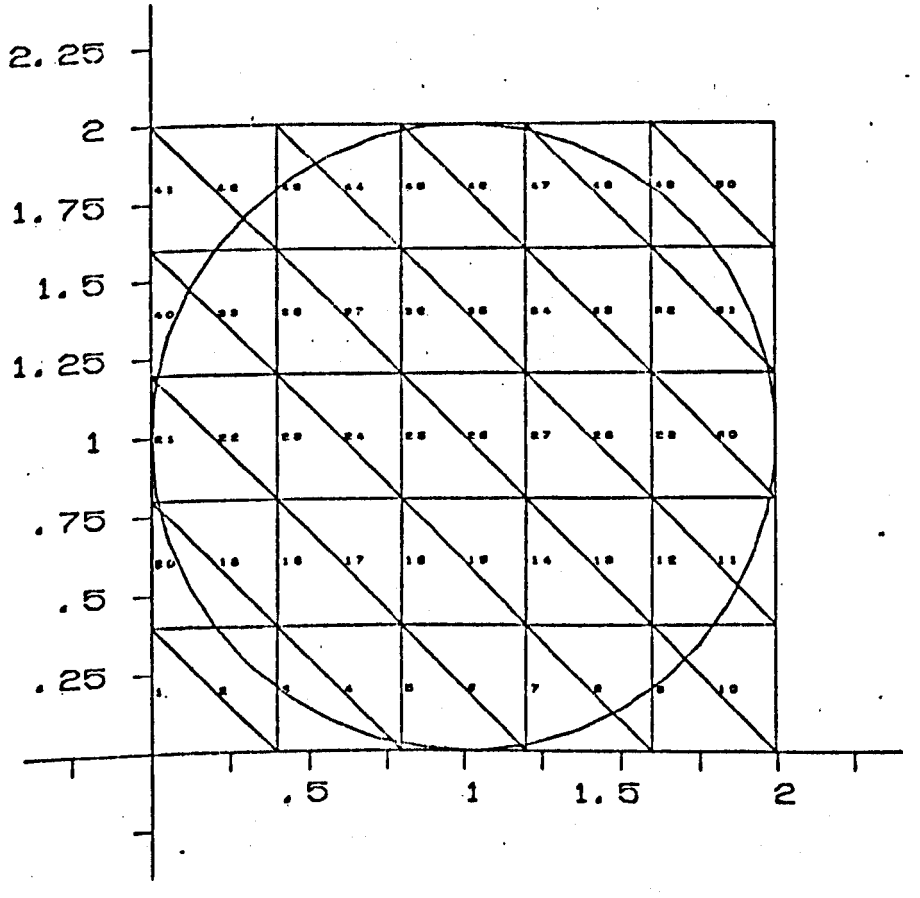


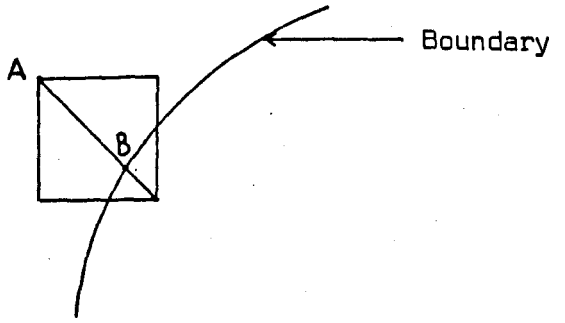
Figure 6.5.

When all the external element numbers have been typed, followed by the terminating number zero, these elements are automatically removed from the displayed region and the options 'Fit Triangles to Region' are again displayed in the instruction area.

It is now necessary to adjust all the elements crossing the boundary by "moving" external vertices or nodes onto the boundary. When key 2 is pressed the following is displayed, together with a tracking cross, replacing the contents of the instruction area:

MOVE CROSS TO EXTERNAL NODE - PRESS KEY 2
THEN MOVE CROSS TO NEAR POSN. ON BOUNDRY
AND PRESS KEY 2

If the external node is common to two triangles, it is usually advisable to move it to the point where the common edge of the triangles cuts the boundary. This avoids creating long thin triangles.



For example, in the diagram above, node A can be moved to point B. Similarly, if the node is common to three triangles it can be moved to some point inside the inner triangle. For example, in the diagram below, node A can be moved to some point on the boundary between points B and C.

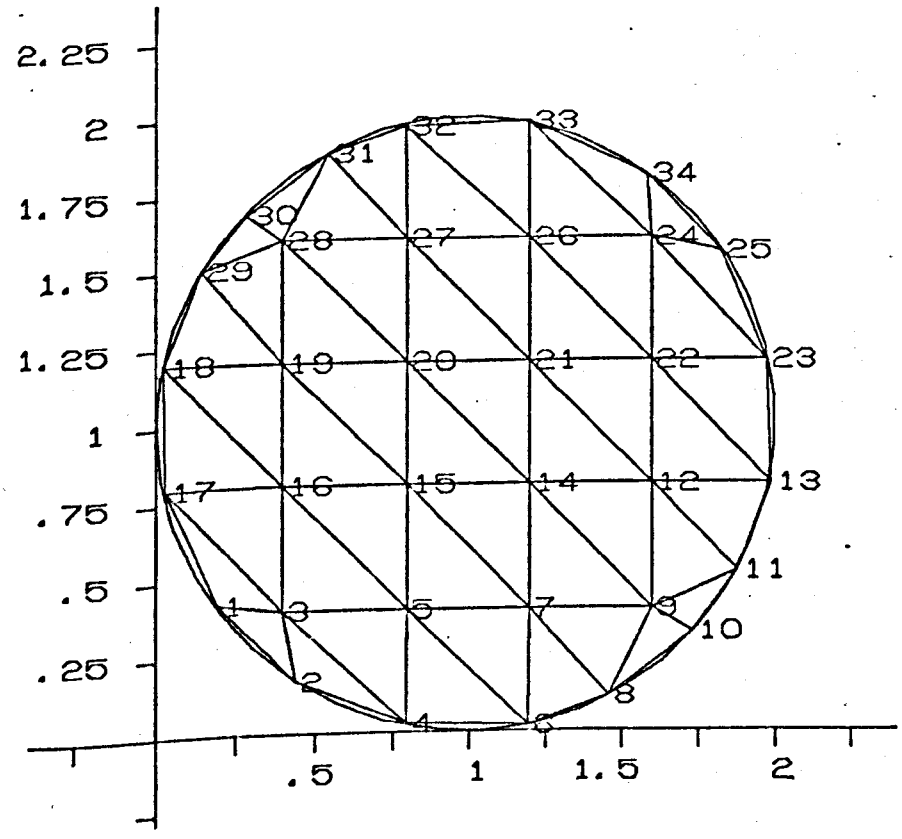
$$\text{TO SOLVE} \quad -\frac{D}{DX}\left(\frac{DU}{DX}\right) - \frac{D}{DY}\left(\frac{DU}{DY}\right) + G*U = F$$

REGION NAME -

CIRCLER1

KEY 1) CONTINUE

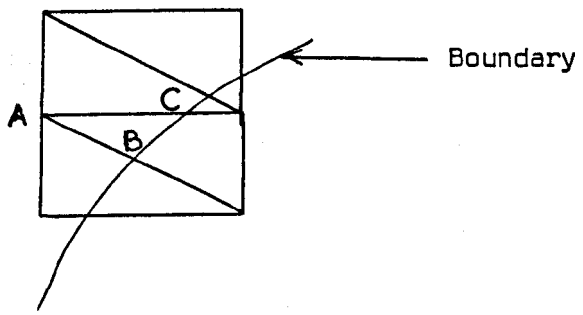
2) PLOT



G = 2

F = 0

Figure 6.6.



When all the external nodes have been moved onto the boundary the user may wish to halve large triangles or those located at a particular point of interest. To carry out this, key 4 is selected and the following instructions are then displayed in the instruction area. A tracking cross is also output on the screen.

MOVE CROSS TO ONE END OF EDGE TO
BE HALVED AND PRESS KEY 1
REPEAT FOR OTHER END

Lines are then drawn automatically from the mid-point of the selected edge to the opposite vertices of the adjacent triangles.

The display file can be plotted onto the digital plotter using key 6. When triangulation is satisfactorily completed key 3 transfers control to the next stage where the user is required to specify the equation to be solved.

During the whole of the program run the following title is displayed at the top of the screen, as shown in Figure 6.6.

$$\text{TO SOLVE} - \frac{D(DU)}{DX(DX)} - \frac{D(DU)}{DY(DY)} + G*U = F$$

where $G \equiv g$

and $F \equiv f$ as defined in equation 4.2 in Chapter 4.

The following instructions replace the contents of the working area:

```
TYPE IN HOMOGENEOUS CONSTANT 'F'
```

and, when the value of F has been supplied, the instruction is replaced by:

```
TYPE EQN. COEFF. 'G'
```

When the value of G has been typed on the teletype, the instruction is replaced by a display of the values assigned to G and F and by the triangulated region of solution with each node numbered. The display file can, if required, be plotted on the digital plotter before continuing (see Figure 6.6).

When the 'continue' key is selected the following options are displayed in the instruction area, above the triangulated region:

```
DEFINE BOUNDARY CONDITIONS
KEY 1)  CONSTANT
    2)  POINT-BY-POINT CONSTANT
    3)  DERIVATIVE U_N + P*U=Q (U_N = NORMAL DERIV)
    4)  COMBINATION OF 2) AND 3)
```

Key 1 is selected when all the boundary nodes are to be assigned the same value. After pressing key 1 the instructions above are replaced by:

```
TYPE CONSTANT B'NDRY VALUE - UB (F0.0)
THEN ALL B'NDRY NODE NUMBERS     (I0)
** TO CORRECT ERROR TYPE -1 AND RESTART **
        ** TO END TYPE ZERO **
```

Alternatively, when the boundary nodes take different values key 2 is used and the instructions are:

```
TYPE B'NDRY NODE NO. + VALUE AT NODE
FOR ALL B'NDRY NODES  (I0, F0.0)
** TO CORRECT TYPE -1 AND RESTART **
      ** TO END TYPE ZERO **
```

When the boundary conditions are of the form $\frac{\partial u}{\partial n} + Pu = Q$ key 3 is selected and the following instructions appear in the instruction area:

```
TYPE P,Q FOR UN + P*U = Q  (2FO.O)
THEN ALL B'NDRY NODE NUMBERS  (IO)
** TO CORRECT ERROR TYPE -1 AND RESTART **
        ** TO END TYPE ZERO **
```

The instructions following the selection of the fourth option (using key 4) are:

```
TYPE NODE NO. + NCB  (2IO)
(NCB = O FOR CONSTANT, = 1 FOR DERIVATIVE
                           CONDITION)
THEN TYPE CONSTANT  (FO.O) OR
     TYPE P,Q FOR UN + P*U = Q  (2FO.O)
** TO CORRECT ERROR TYPE -1 AND RESTART **
        ** TO END TYPE ZERO **
```

When the description of the boundary conditions is complete control is automatically transferred to the computation level.

6.5.3. Computation Level

The solution is carried out automatically and the solution surface is then displayed, in the working area, as a projection onto the plane of the screen. Control moves automatically to the finalisation level.

6.5.4. Finalisation Level

A set of options, the 'end' options are displayed with the solution surface. These are:

$$\text{TO SOLVE} \quad -\frac{D}{DX}\frac{(DU)}{(DX)} - \frac{D}{DY}\frac{(DU)}{(DY)} + G*U = F$$

END OPTIONS

KEY 1) ROTATE IN XY PLANE-TYPE NO. DEGS
(TYPE 0 FOR ORIGINAL VIEW)

2) PLOT

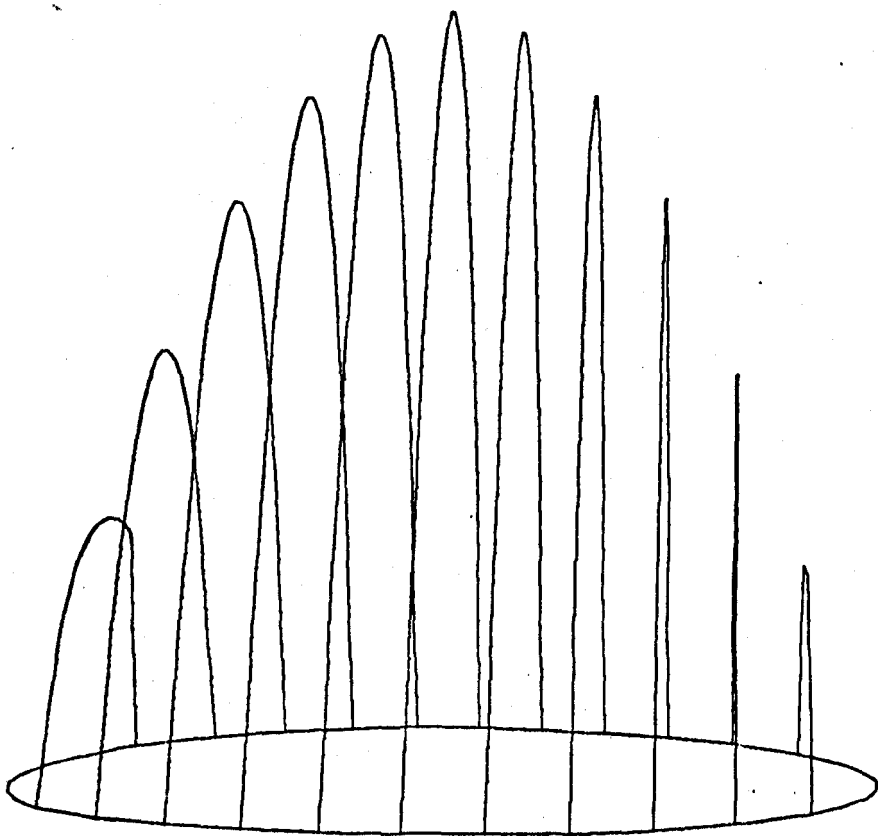
3) MOVE TO NEXT DATA SET

4) RESCALE

8) STOP

REGION NAME -

CIRCLER1



0 <= X <= 1.9966

.00088 <= Y <= 1.9991

0 <= U <= .4811

Figure 6.7.

END OPTIONS

KEY 1) MOVE TO NEXT DATA SET

2) PLOT

3) ROTATE XY PLANE - TYPE NO. DEGS (IO)

(TYPE 0 FOR ORIGINAL VIEW)

4) RESCALE

5) DISPLAY CONTOUR PLOT

8) STOP

An example of the display file at this stage is given in Figure 6.7.

To rescale the u-axis of the projected surface key 4 is pressed and the following instruction appears in the instruction area:

TYPE UMIN AND UMAX

When the new axis limits have been supplied control returns to the 'end' options.

To obtain a contour plot of the solution surface key 5 is activated and the options are then replaced by:

TYPE NO. CONTOURS (IO)
TYPE VALUES OF CONTOURS (10F0.0)

When all the values are supplied the 'end' options are again available but choices 3), 4) and 5) are replaced by:

6) DISPLAY PROJECTION OF SOLN.

Some of the facilities available in FESOLV are illustrated by an example problem in Chapter 7.

7. APPLICATIONS OF THE INTERACTIVE GRAPHICS SYSTEM FOR DIFFERENTIAL EQUATIONS

To illustrate the three programs GIDODE, GIDPDE and FESOLV described in the previous chapter, an example problem for each program is examined. It is not feasible to illustrate all the facilities available but the applications show some of the most useful aspects of interactive graphics in the field of differential equations.

Simple, but non-trivial examples are selected since the purpose of the chapter is to illustrate the uses of the system rather than to find the solutions to difficult or large problems.

7.1. Ordinary Differential Equations

The solution of a problem, consisting of a set of three first-order ordinary differential equations, is described in this section. The example is taken from a report by H. H. Robertson[50] on the integration of systems of stiff ordinary differential equations. The equations are

$$y_1^1 = -0.04y_1 + 10^4 y_2 y_3$$

$$y_2^1 = 0.04y_1 - 10^4 y_2 y_3 - 3 \times 10^7 y_2^2$$

$$y_3^1 = 3 \times 10^7 y_2^2$$

to be integrated from x = 0 to x = 40, using the initial values $y_1 = 1$ and $y_2 = y_3 = 0$ at x = 0. Initially an error (EPS) of 0.01 is required. This is an absolute control if y_i (i = 1, 2, 3) is decreasing and a relative control if y_i (i = 1, 2, 3) is increasing.

Since only five data records are required (see Chapter 6.3.1 for a description of the data) it is easily possible to type them in at run time

rather than prepare them on cards to be loaded with the program. The equation definition, however, is less simple so that it is advisable to prepare a user function before run time and load it with the program. The function must be called DERG and is as follows:

```
SUBROUTINE DERG(F,Y,X)
DIMENSION F(3), Y(3)
F(1) = -0.04*Y(1) + 1.0E4*Y(2)*Y(3)
F(2) = -F(1) -3.0E7*Y(2)*Y(2)
F(3) = -F(1) -F(2)
RETURN
END
```

When the subroutine DERG has been prepared it is loaded together with the control cards for GIDODE as described in Chapter 3.3.

The first choice the user must make is the method of solution. The instructions for this and the following choices in level 1 are fully described in Chapter 6.3.2. In this case key 1 is pressed to select Adams' method of solution for non-stiff initial value problems. It is assumed here that the user does not know that the equations are stiff. These instructions are next replaced by those for the selection of the amount of lineprinter output. 'Complete' output is chosen by pressing key 3. The selection of input devices for the equations and the data must next be made. For this problem, key 2 is pressed first to select the user function, via the card reader as the means of supplying the equations, followed by key 1 to choose the teletype for the input of data. It is now necessary to input the data. A list of variable names and formats for each record is displayed, together with instructions for typing the data. The values typed on the console are:

```
0.0, 0.1
 3, 0.01
1.0
0.0
0.0
```

An initial maximum step length of H = 0.1 is used. The initialising level ends with a summary of the problem and the 'begin' options display on the screen. Key 4 is selected and the display file is drawn onto the digital plotter. Figure 7.1 is a copy of this plot.

Provided no errors have been made in the input of data, key 1 is pressed and the computation level, level 2, is entered.

Before computation actually begins it is necessary to select the variables to be displayed numerically and graphically. The instructions described in Chapter 6.3.3 are displayed in the working area of the screen. In each case key 7 is pressed to select the default values which, for numerical display are x, y_1 and e_1 (the error in y_1), and for graphical display are the two graphs, x - y_1 and x - e_1. If required these decisions can be changed at a later stage in the program.

The following instruction now replaces those in the working area:

TYPE BOUNDS FOR X AXIS

The initial limits selected are 0.0 and 40.0, so these values are now typed on the console. Similar instructions are next displayed for the Y1 (y_1) axis and the E1 (e_1) axis. The bounds supplied are 1.0 and 0.5 for Y1 and 0 and 0.01 for E1. These bounds can be changed at any time during the computation stage by rescaling axes.

At this point in the program computation actually begins. The solution is evaluated for one step along x and output together with the 'run' options. In this case however, Adams' method fails to converge and the message 'H TOO LARGE' is displayed on the screen. The display file is drawn onto the digital plotter and shown in Figure 7.2. To change the value of HMAX key 2 is pressed, and the new value, in this case 0.001, is typed on the

SOLUTION OF INITIAL-VALUE O.D.E. USING ADAMS' METHOD

TO SOLVE 3 SIMULTANEOUS EQUATIONS

WHERE X(1) = 0 HMAX= 1.-1

Y1 (1) = 1.0

Y2 (1) = 0

Y3 (1) = 0

EPS= 1.-2

BEGIN OPTIONS

KEY 1) CONTINUE

2) CHANGE DATA RECORD OR EQ'NS

3) MOVE TO END OPTIONS

4) PLOT TO DIGITAL PLTER

8) STOP

Figure 7.1.

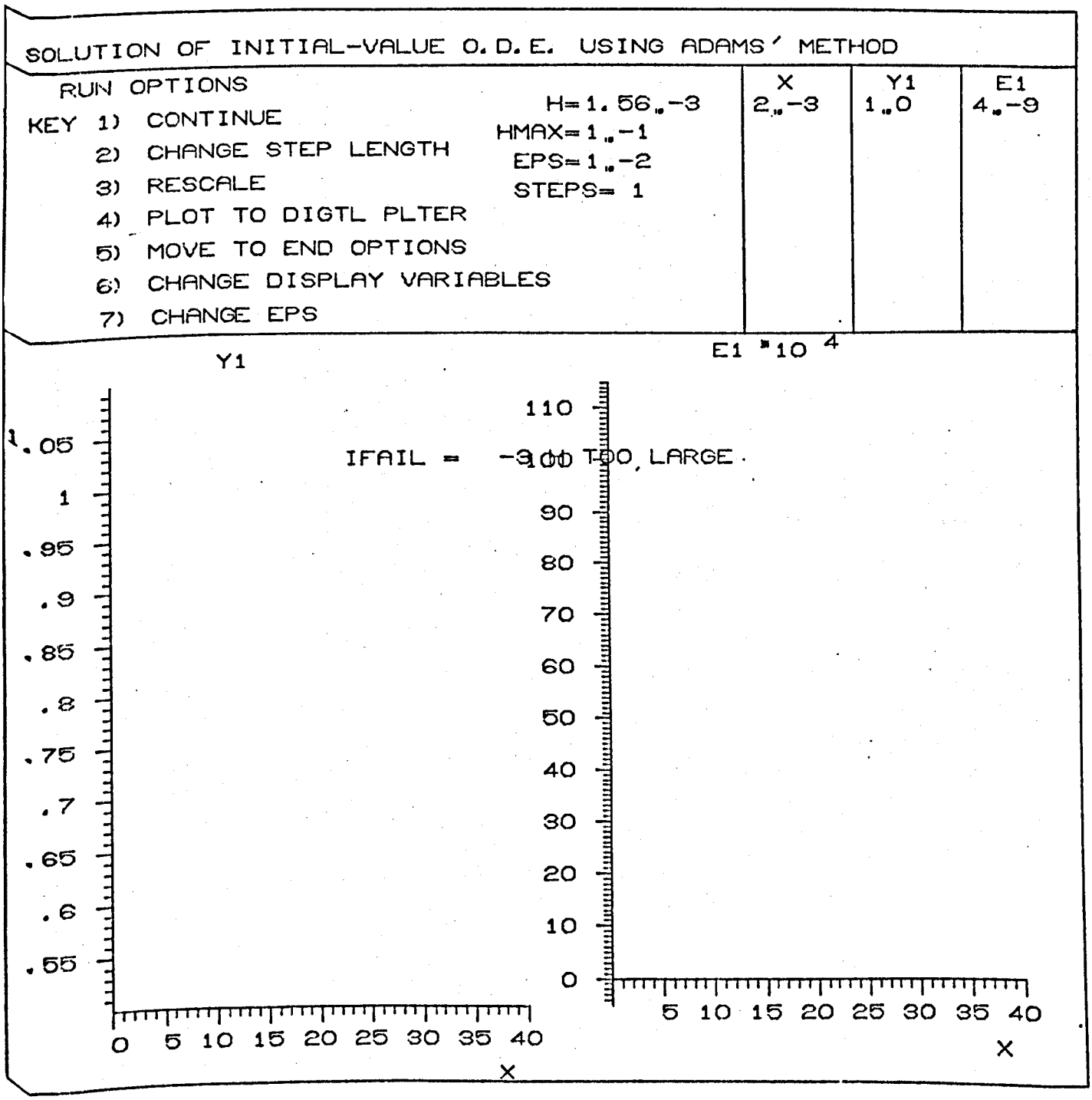

SOLUTION OF INITIAL-VALUE O.D.E. USING ADAMS' METHOD
RUN OPTIONS
KEY 1) CONTINUE
2) CHANGE STEP LENGTH
3) RESCALE
4) PLOT TO DIGTL PLTER
5) MOVE TO END OPTIONS
6) CHANGE DISPLAY VARIABLES
7) CHANGE EPS
H=1.56..-3
HMAX=1..-1
EPS=1..-2
STEPS= 1
X
2..-3
Y1
1..0
E1
4..-9
Y1
E1 *10 4
IFAIL = -3 00 TOO LARGE.
1.05
1
.95
.9
.85
.8
.75
.7
.65
.6
.55
0 5 10 15 20 25 30 35 40
X
110
100
90
80
70
60
50
40
30
20
10
0
5 10 15 20 25 30 35 40
X

Figure 7.2.

teletype. Key 1 is then pressed and the computation continues using the new value for the maximum step size. (The minimum possible step size is HMAX x 10^{-2}). Convergence, however, is still not achieved. This means that the actual step length required must be less than 0.001×10^{-2} so, assuming that the system of equations is 'stiff' Gear's method of solution is tried. It is now necessary to change the method of solution and to change the maximum step size back to its initial value of 0.1. These two changes could be performed in either order. We choose to change the step size first since this can be carried out at the current level. In the same way as H was reduced above, key 2 is pressed and the new value, in this case 0.1, is typed on the teletype. The method of solution is now changed by first pressing key 5 to move to the 'end' options. Key 2 is then selected and the options, described in Chapter 6.3.4, which include changing the method of solution, are displayed. As instructed the number 200 is typed on the teletype and control reverts to the initialisation level where the selection of the method of solution is available. When the new choice (key 3) has been made the problem definition and the 'begin' options are again displayed. The display is very similar to that shown in Figure 7.1, the only difference being that 'Adams' method' in the title is now replaced by 'Gear's method'. Since no further changes are required key 1 is pressed to transfer control to the computation level.

Instructions for selecting the display variables are again displayed. These can remain unchanged so key 6 is pressed (for 'unchanged' choice) for both numerical and graphical display variables. An unchanged choice of graphs means that it is not necessary to redefine the axes limits.

Computation now begins using Gear's method and the solution after one step is output together with the 'run' options. It is now possible to raise

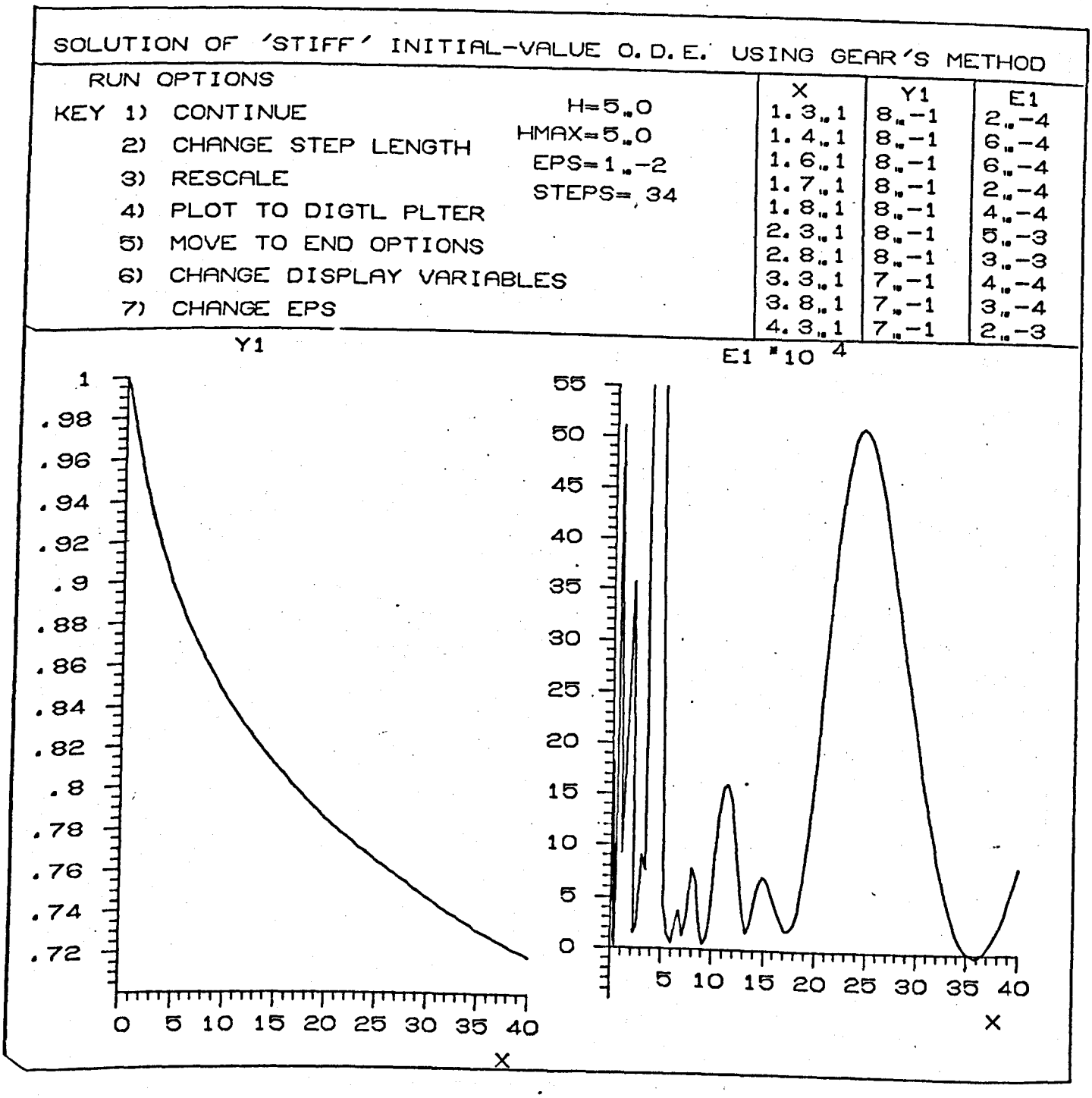

SOLUTION OF 'STIFF' INITIAL-VALUE O.D.E. USING GEAR'S METHOD
RUN OPTIONS
KEY 1) CONTINUE
2) CHANGE STEP LENGTH
3) RESCALE
4) PLOT TO DIGTL PLTER
5) MOVE TO END OPTIONS
6) CHANGE DISPLAY VARIABLES
7) CHANGE EPS
H=5.0
HMAX=5.0
EPS=1.-2
STEPS= 34
X Y1 E1
1.3.1 8.-1 2.-4
1.4.1 8.-1 6.-4
1.6.1 8.-1 6.-4
1.7.1 8.-1 2.-4
1.8.1 8.-1 4.-4
2.3.1 8.-1 5.-3
2.8.1 8.-1 3.-3
3.3.1 7.-1 4.-4
3.8.1 7.-1 3.-4
4.3.1 7.-1 2.-3
Y1
1 .98 .96 .94 .92 .9 .88 .86 .84 .82 .8 .78 .76 .74 .72
0 5 10 15 20 25 30 35 40
X
E1 *10 4
55 50 45 40 35 30 25 20 15 10 5 0
5 10 15 20 25 30 35 40
X

Figure 7.3.

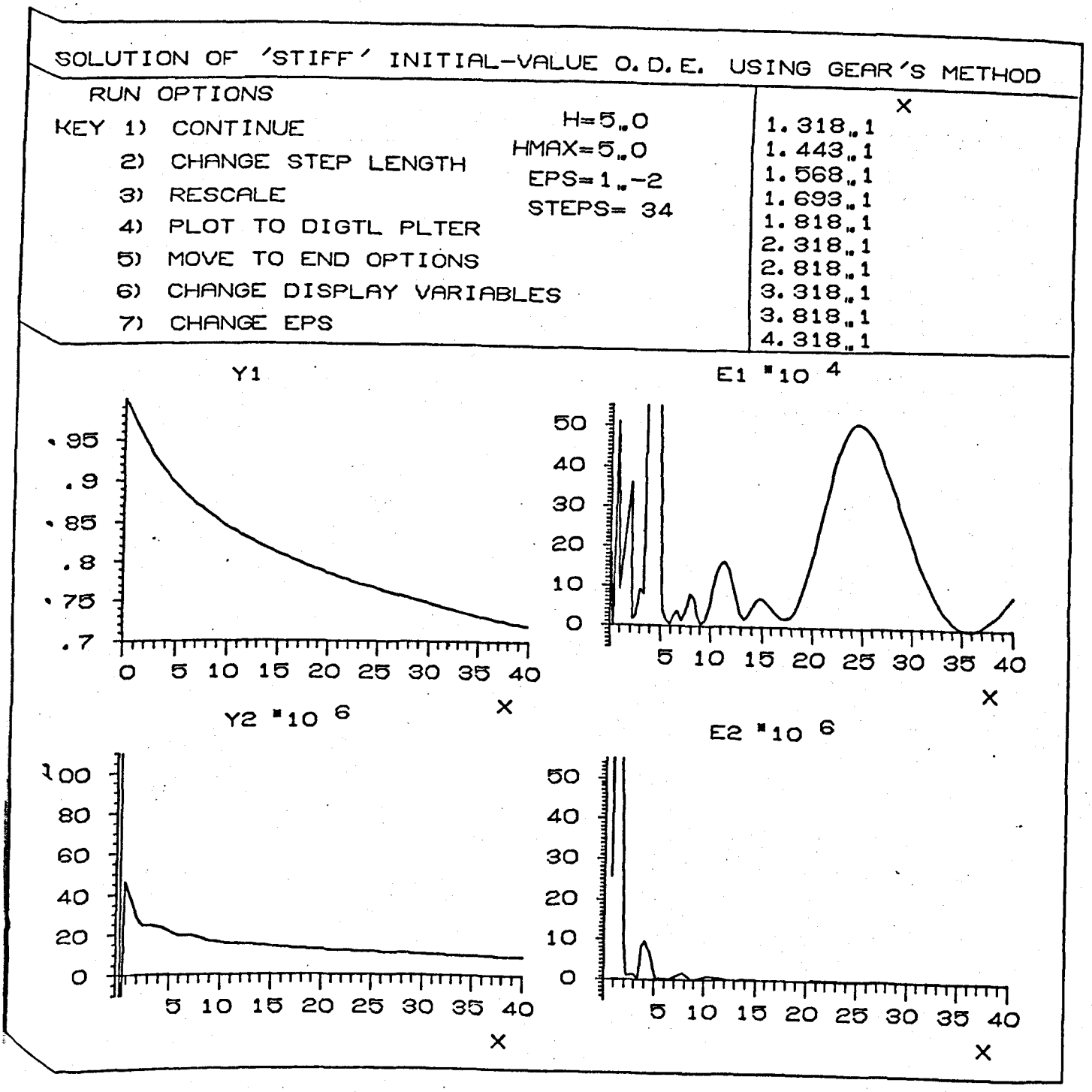

SOLUTION OF 'STIFF' INITIAL-VALUE O.D.E. USING GEAR'S METHOD
RUN OPTIONS
KEY 1) CONTINUE
2) CHANGE STEP LENGTH
3) RESCALE
4) PLOT TO DIGTL PLTER
5) MOVE TO END OPTIONS
6) CHANGE DISPLAY VARIABLES
7) CHANGE EPS
H=5.0
HMAX=5.0
EPS=1.-2
STEPS= 34
X
1.318.1
1.443.1
1.568.1
1.693.1
1.818.1
2.318.1
2.818.1
3.318.1
3.818.1
4.318.1
Y1
.95
.9
.85
.8
.75
.7
0 5 10 15 20 25 30 35 40
X
E1 *10 4
50
40
30
20
10
0
5 10 15 20 25 30 35 40
X
Y2 *10 6
100
80
60
40
20
0
5 10 15 20 25 30 35 40
X
E2 *10 6
50
40
30
20
10
0
5 10 15 20 25 30 35 40
X

Figure 7.4.

SOLUTION OF 'STIFF' INITIAL-VALUE O.D.E. USING GEAR'S METHOD

RUN OPTIONS

KEY 1) CONTINUE
2) CHANGE STEP LENGTH
3) RESCALE
4) PLOT TO DIGTL PLTER
5) MOVE TO END OPTIONS
6) CHANGE DISPLAY VARIABLES
7) CHANGE EPS

H=4.854.0
HMAX=5.0
EPS=1.-3
STEPS= 51

X	Y1	E1
1.1.1	8.-1	3.-4
1.3.1	8.-1	6.-4
1.5.1	8.-1	4.-4
1.8.1	8.-1	8.-4
2.1	8.-1	7.-4
2.3.1	8.-1	5.-4
2.6.1	8.-1	3.-4
3.1.1	7.-1	6.-4
3.6.1	7.-1	7.-4
4.1.1	7.-1	5.-4

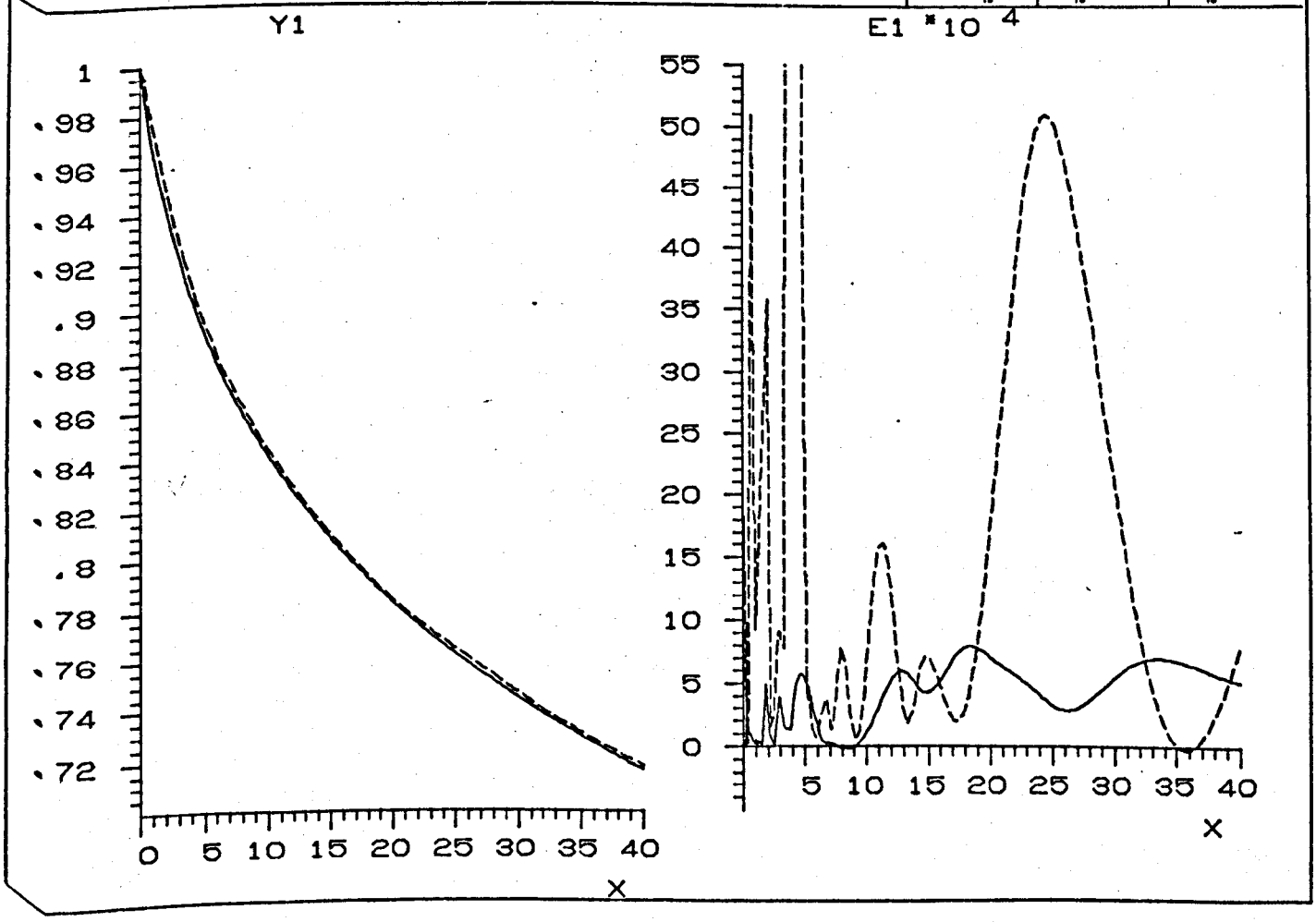


Figure 7.5.

key 1 into the fixed "ON" position so that computation continues, step by step until key 1 is switched "OFF". If at any time a curve generated on the screen is beyond the limits of an axis, the axis can be rescaled by switching key 1 "OFF" and pressing key 3. The new limits for the axis selected for rescaling are typed on the console. In addition, if, during the computation, the actual step size used (called H on the screen) becomes close, to the maximum permitted step size (called HMAX) it may be desirable to increase HMAX, to speed up the solution, by the method previously described.

When the value of x has reached 40, key 1 is switched "OFF" and the display file is plotted (key 4). This is shown in Figure 7.3.

When key 6 is pressed the display variables can be changed. Figure 7.4 shows four graphs of $x - y_1$, $x - e_1$, $x - y_2$ and $x - y_3$, and more accurate values of the independent variable x.

It is possible to 'save' the current solution and compare it to the solution of a perturbed problem. We shall consider the same problem with the more stringent error control of EPS = 0.001. The first step is to press key 7 and type in the new value of EPS. This is followed by key 5 which transfers control to the finalisation stage and the END options. Key 3 now saves the current solution and the instructions for changing data or equations are displayed on the screen. As requested, '20' is now typed on the teletype and control reverts to the initialisation level and the problem definition and BEGIN options are again displayed. The solution is evaluated in the same way as previously described. The saved solutions and errors are plotted on the displayed graphs as dashed lines. When the solution has been evaluated up to x = 40 the display file is plotted onto the digital plotter (Figure 7.5). It can be seen that the error in y_1 (e_1) is reduced

by approximately a factor of ten but the number of steps taken is increased from 34 to 51.

To terminate the program the finalisation level is entered by pressing key 5 and the program is then stopped using key 8.

7.2. Partial Differential Equations

There are two programs, GIDPDE and FESOLV available to solve partial differential equations. They use finite differences and a finite element method of solution respectively. One example will be given for each program.

7.2.1. Finite Difference Method of Solution

For a one-dimensional problem, solutions to similar problems can again be displayed simultaneously as described for ordinary differential equations. However, a two-dimensional elliptic equation is used as the example here since it illustrates some of the difficulties associated with a two-dimensional rather than a one-dimensional problem as well as those associated with an iterative method of solution. The method of solution used is the Peaceman-Rachford alternating direction method, as described in Appendix B.

The equation selected is Laplace's equation defined on a rectangle. This is

$$\frac{\partial u^2}{\partial x^2} + \frac{\partial u^2}{\partial y^2} = 0 \qquad \text{for } 0 \leqslant x \leqslant 1, \quad 0 \leqslant y \leqslant 1$$

with the boundary conditions

$$u(x,0) = 1$$

$$\frac{\partial u(x,1)}{\partial y} = 0$$

$$\frac{\partial u(0,y)}{\partial x} = 0$$

$$\frac{\partial u(1,y)}{\partial x} + 0.1\ u(1,y) = 0$$

The starting value used is u(x,y) = 1. We shall choose to use a constant acceleration parameter, that is, the same value at each iteration. It is known that for constant boundary values the optimum parameter value is approximately 0.6 (see page 152 of (52)). This value is, therefore, used as an initial estimate.

The data is provided on cards which are loaded with the program control cards. All other information (such as the equation description) is provided during the program run using the keyboard, light pen or sense keys. It would, of course, be possible to describe the problem entirely at run time. The format of the data records is given in Chapter 6.4.1, and the actual values supplied are as follows: (∇ denotes a space)

∇2	(Insulated boundary at y = 1)
∇1∇1.0	(Constant boundary value of 1.0 at y = 0)
∇2	(Insulated boundary at x = 0)
∇3∇0.1	(Radiative boundary with h = 0.1 at x = 1)
0.0∇∇∇∇∇∇∇1.0∇∇∇∇∇∇∇0.0∇∇∇∇∇∇∇1.0	(Region limits)
1111∇0.6	(Number of mesh points N = 11, M = 11, acceleration parameter = 0.6)
1.0	(Starting value, u(x,y) = 1.0)

The control cards described in Chapter 3.3 are loaded together with the data cards listed above.

The initialisation level begins with the display, in the working area, of the instructions for selecting the amount of lineprinter output. Key 3

is pressed to select 'complete' lineprinter output which will include the problem description and the solution values at each iteration. The instructions are then replaced by those headed 'Select type of P.D.E.' (see Chapter 6.4.2) and as an elliptic equation is to be solved key 3 is pressed. It is next necessary to define the coefficients A_1, A_2 ... A_6 given in equation 4.4. The appropriate instructions are displayed, replacing those already in the working area, for each coefficient in turn. The default value 1.0 is selected for A_1 and A_2 (using key 2) and the remaining coefficients are given the value 0.0 by pressing key 1 for each.

The next set of instructions displayed are those requesting the user to select the data input device. The data, in this case, is supplied on cards, therefore key 2 is pressed to select the card reader as the input device.

At this stage an identification area is set up at the top of the screen in which the following title is displayed:

SOLUTION OF LINEAR ELLIPTIC P.D.E. IN TWO SPACE DIMENSIONS

This title remains until the type of equation is changed. The instructions in the working area are replaced by a summary of the problem together with the 'begin' options. The contents of the display file is, at this point, drawn onto a digital plotter by selecting key 4. This plot is shown in Figure 7.6. Since the problem description is correct, control is transferred to level two, the computation level, by pressing key 1.

One iteration is carried out automatically after which the solution surface and 'run' options replace the problem description and 'begin' options in the working area. The solution surface at this stage shows signs of instability but a further iteration is carried out (by pressing key 1 to

SOLUTION OF LINEAR ELLIPTIC P.D.E. IN TWO SPACE DIMENSIONS

TO SOLVE

$$0 = A1\frac{D}{DX}\left(\frac{DU}{DX}\right) + A2\frac{D}{DY}\left(\frac{DU}{DY}\right) + A3\frac{DU}{DX} + A4\frac{DU}{DY} + A5*U + A6$$

WHERE

A1= 1 A2= 1 A3= 0

A4= 0 A5= 0 A6= 0

BOUNDARY CONDITIONS

BNDRY NO.

1 DU/DN=0

2 U= 1

3 DU/DN=0

4 DU/DN= -.1 * (UB- 0)

GRID DETAILS

XMIN= 0 XMAX= 1 DX=.1

YMIN= 0 YMAX= 1 DY=.1

BEGIN OPTIONS

KEY 1) CONTINUE

2) CHANGE DATA RECORD OR COEFFS

3) MOVE TO END OPTIONS

4) PLOT TO DIGITAL PLTER

8) STOP

Figure 7.6.

confirm this). A copy of the display file is obtained on the digital plotter (Figure 7.7) by pressing key 4.

The acceleration parameter is changed by pressing key 2 and then typing the new value on the keyboard. The solution can then be restarted by moving to the 'end' options and pressing key 3. The necessary instructions at each stage are displayed on the screen. The new acceleration parameter used is 0.9, but from Figure 7.8 it can be seen that this also gives a divergence solution. The next value used for the acceleration parameter is 1.0 which gives convergence.

Computation is allowed to continue for successive iterations by raising key 1 so that it remains in the 'on' position. At intervals during the computation key 1 is switched off so that the surface being displayed can be rescaled. Key 3 is pressed and the instructions

TYPE UMIN AND UMAX

are displayed. The final limits typed are .95 and 1.0.

Iteration is terminated when the maximum percentage change in a grid value is less than 0.1 for one iteration. This occurs after 20 iterations. The display file is then plotted onto the digital plotter (Figure 7.9) by pressing key 4. In order to rotate the solution surface key 6 is pressed and the required number of degrees of rotation about the U-axis typed on the keyboard. Figure 7.10 shows the surface rotated through 235^{o}.

Control is transferred to the finalisation level, by pressing key 5, where the 'end' options are available and are displayed on the screen. Figure 7.11 shows the error surface obtained by pressing key 4 and rotating the surface through 235^{o}.

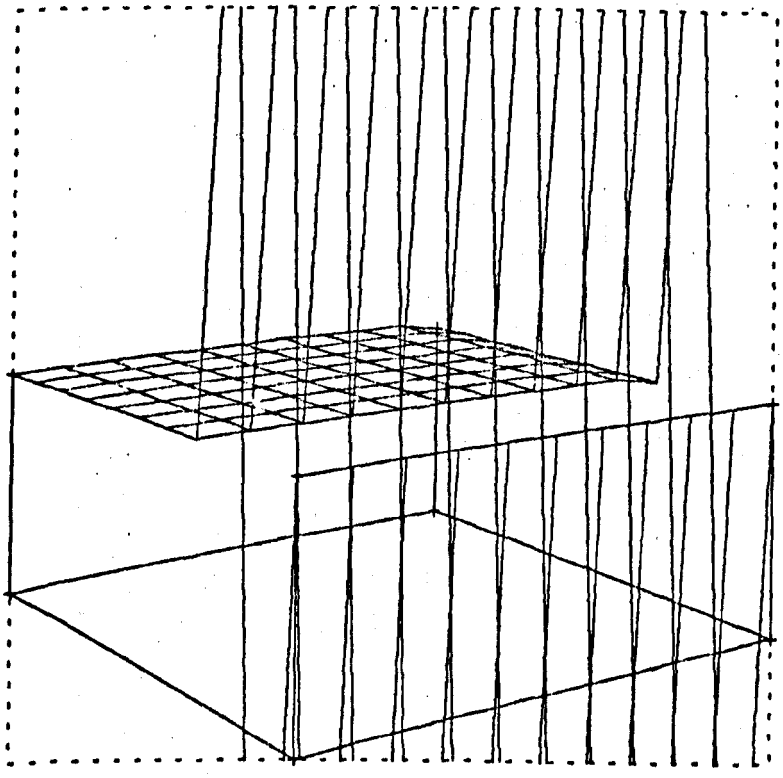
At this point in the program the facility for changing the data or equation coefficients before either continuing or restarting the solution

SOLUTION OF LINEAR ELLIPTIC P.D.E. IN TWO SPACE DIMENSIONS

RUN OPTIONS

KEY 1) CONTINUE
2) CHANGE ACCN. PRMTR
3) RESCALE
4) PLOT TO DIGTL PLTER
5) MOVE TO END OPTIONS
6) ROTATE-TYPE NO. DEGS (IO)

ITN. NO.	MAX % DIFF
1	166.67
2	277.78



0 <= X <= 1

0 <= Y <= 1

0 <= U <= 2.6667

ACCN. PARAMETER

.6

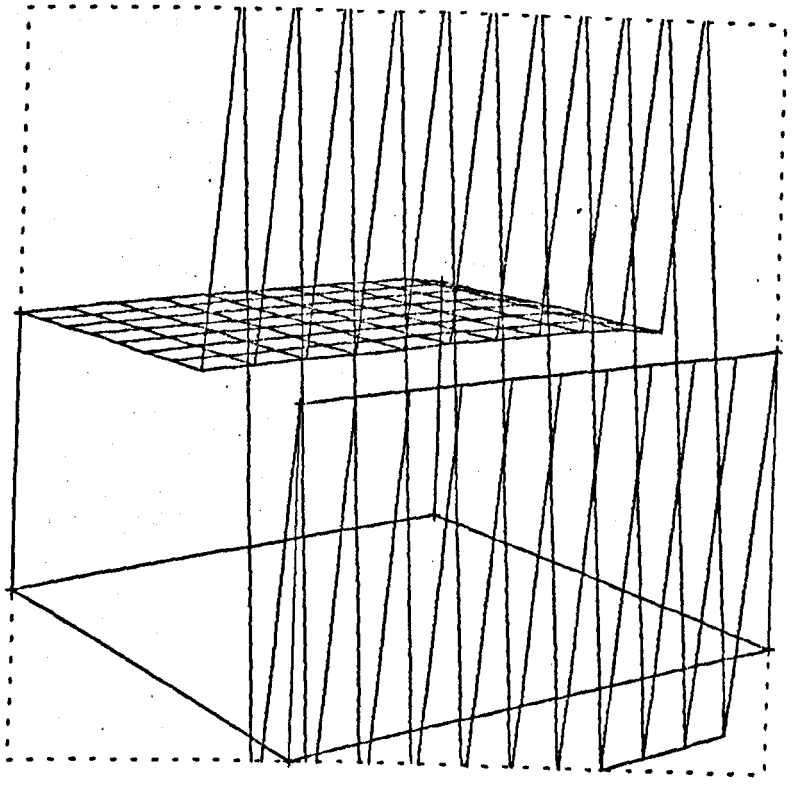
Figure 7.7.

SOLUTION OF LINEAR ELLIPTIC P.D.E. IN TWO SPACE DIMENSIONS

RUN OPTIONS

KEY 1) CONTINUE
2) CHANGE ACCN. PRMTR
3) RESCALE
4) PLOT TO DIGTL PLTER
5) MOVE TO END OPTIONS
6) ROTATE-TYPE NO. DEGS (IO)

ITN. NO.	MAX % DIFF
1	111.11
2	123.46



0 <= X <= 1

0 <= Y <= 1

0 <= U <= 2.1111

ACCN. PARAMETER

.9

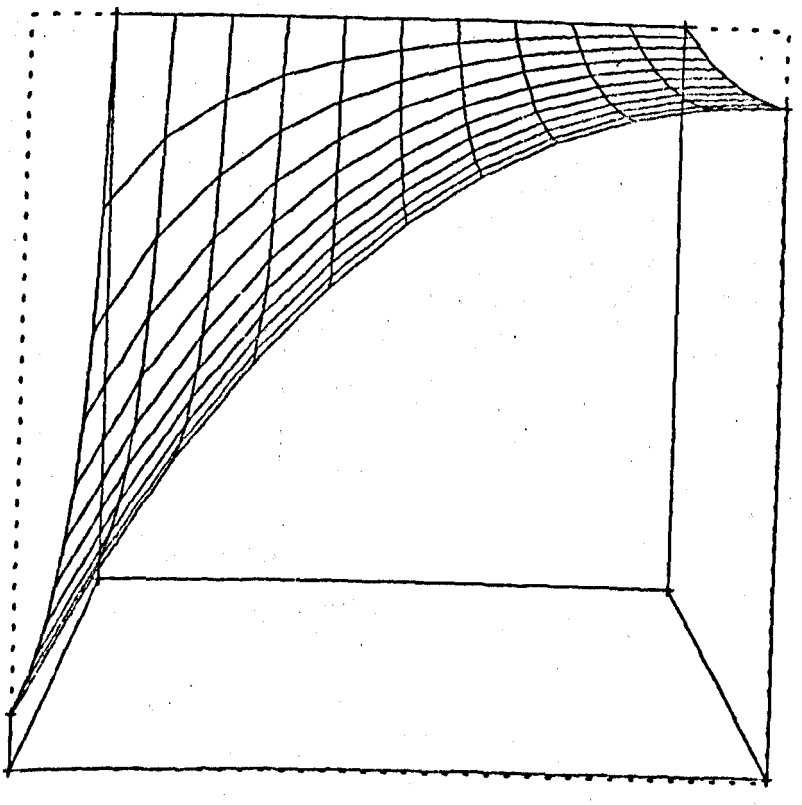
Figure 7.8.

SOLUTION OF LINEAR ELLIPTIC P.D.E. IN TWO SPACE DIMENSIONS

RUN OPTIONS

KEY 1) CONTINUE
2) CHANGE ACCN. PRMTR
3) RESCALE
4) PLOT TO DIGTL PLTER
5) MOVE TO END OPTIONS
6) ROTATE-TYPE NO. DEGS (IO)

ITN.	NO. MAX % DIFF
11	.1927
12	.14578
13	.14058
14	.14013
15	.13462
16	.11463
17	.10982
18	.10863
19	.10388
20	.09404



0 <= X <= 1

0 <= Y <= 1

.95 <= U <= 1

ACCN. PARAMETER

1

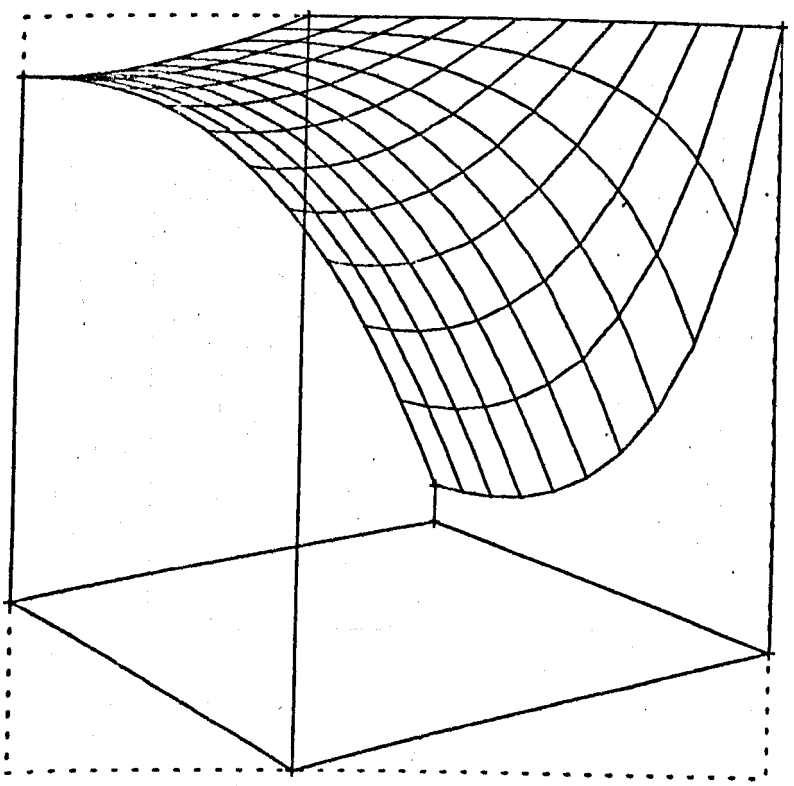
Figure 7.9.

SOLUTION OF LINEAR ELLIPTIC P.D.E. IN TWO SPACE DIMENSIONS

RUN OPTIONS

KEY 1) CONTINUE
2) CHANGE ACCN. PRMTR
3) RESCALE
4) PLOT TO DIGTL PLTER
5) MOVE TO END OPTIONS
6) ROTATE-TYPE NO. DEGS (IO)

ITN.	NO. MAX % DIFF
11	.1927
12	.14578
13	.14058
14	.14013
15	.13462
16	.11463
17	.10982
18	.10863
19	.10388
20	.09404



0 <= X <= 1

0 <= Y <= 1

.95 <= U <= 1

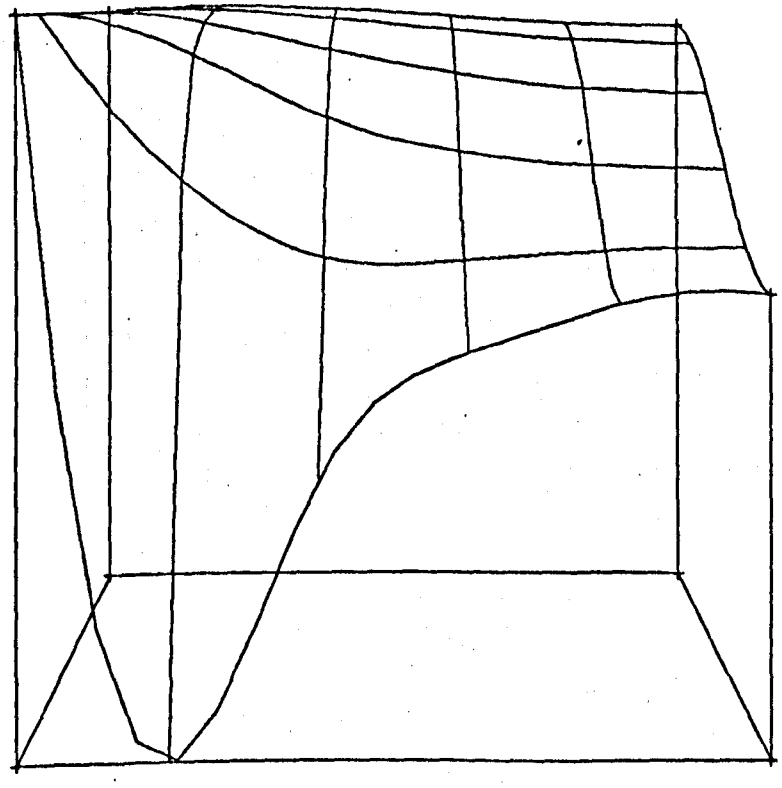
ACCN. PARAMETER

1

Figure 7.10.

SOLUTION OF LINEAR ELLIPTIC P.D.E. IN TWO SPACE DIMENSIONS

PERCENTAGE ERROR



-.0541 < %E < .00032

ESTIMATED MAXIMUM % ERROR IN FINAL TEMPERATURES=.054083924

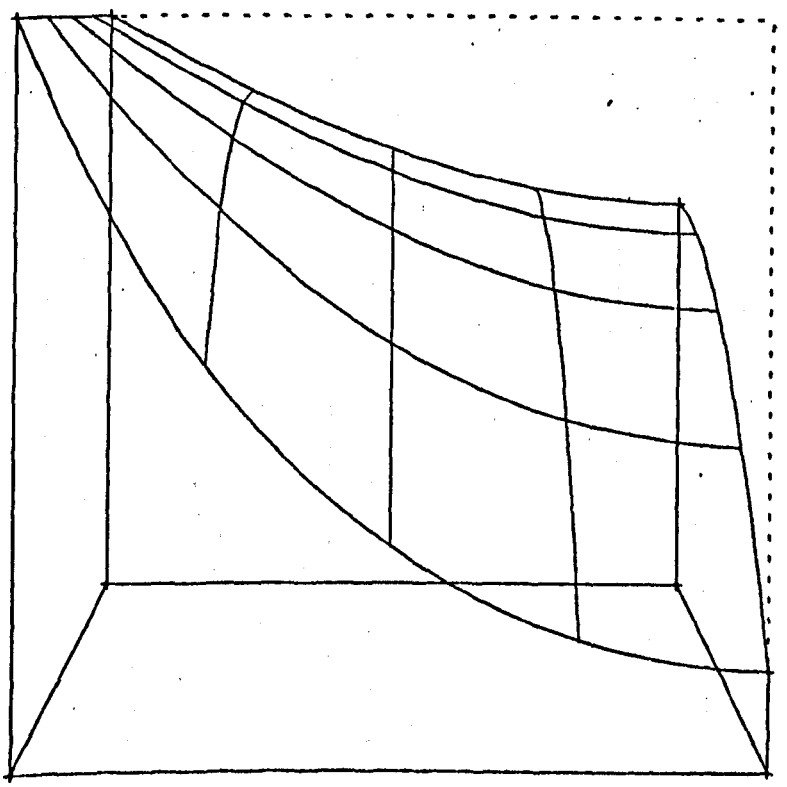
Figure 7.11.

SOLUTION OF LINEAR ELLIPTIC P.D.E. IN TWO SPACE DIMENSIONS

RUN OPTIONS

KEY 1) CONTINUE
2) CHANGE ACCN. PRMTR
3) RESCALE
4) PLOT TO DIGTL PLTER
5) MOVE TO END OPTIONS
6) ROTATE-TYPE NO. DEGS (IO)

ITN.	NO. MAX % DIFF
4	.51479
5	.37528
6	.49097
7	.3637
8	.26515
9	.19585
10	.20288
11	.14923
12	.13284
13	.09812



0 <= X <= 1

0 <= Y <= 1

.92 <= U <= 1

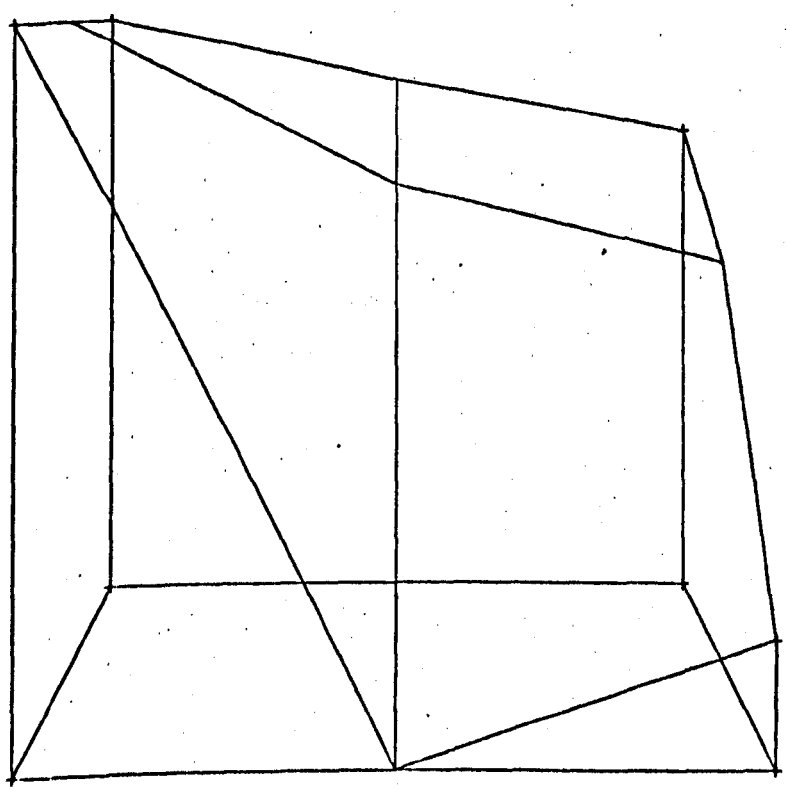
ACCN. PARAMETER

1

Figure 7.12.

SOLUTION OF LINEAR ELLIPTIC P.D.E. IN TWO SPACE DIMENSIONS

PERCENTAGE ERROR



-.1495 < %E < 0

ESTIMATED MAXIMUM % ERROR IN FINAL TEMPERATURES= .14954681

Figure 7.13.

is available. As an illustration the number of grid points is reduced from 11 x 11 to 5 x 5 and the computation restarted. As a first step key 3 is pressed to 'CHANGE DATA OR COEFFS + RERUN'. A numbered list of all the data records is displayed together with the instruction for changing any record. When the new values have been supplied (N = 5, M = 5) the new problem description is displayed with the 'begin' options.

Control is transferred to the computation level using key 1 and the solution is continued in the same way as previously described. The solution surface (rotated through 235^{o}) when the maximum percentage change is less than 0.1 is shown in Figure 7.12. The error surface, again rotated through the same angle is shown in Figure 7.13. The program is finally terminated using key 8.

7.2.2. Finite Element Method of Solution

The example used in this section, for solution using FESOLV is Poisson's Equation defined on a unit circle with zero boundary values. That is:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + 2 = 0$$

with $$r = \sqrt{x^2 + y^2} \leq 1$$

and $$u_b = 0$$

where u_b is the value of u on the boundary.

Although it is possible to define the region of solution using the light pen and tracking cross (in fact the problem can be solved entirely without the use of data cards) greater accuracy is possible by supplying points on the boundary as data.

The data records, with variables and formats are defined in Chapter 6.5.1. For this problem the data will be:

CIRCLER1	Problem name
0.0, 1.0	Starting point for boundary definition
2,	Type of boundary - single closed curve
11, 40	11 points on boundary to be supplied. 40 equally spaced points on the boundary curve to be joined by straight lines.
0.2, 1.6	
0.4, 1.8	
1.0, 2.0	
1.6, 1.8	
1.8, 1.6	
2.0, 1.0	Points on the boundary curve
1.8, 0.4	
1.6, 0.2	
1.0, 0.0	
0.4, 0.2	
0.2, 0.4	
5, 10	Number of triangles along x and y respectively.

When the data has been prepared it is loaded into the computer together with the control cards for FESOLV described in Chapter 3.3.

The first choice to be made on-line is again the amount of output to be sent to the lineprinter. The instructions described in Chapter 6.5.1 are displayed in the working area in the centre of the screen. Key 2 is selected to give lineprinter output of the problem definition and the solution at each node.

These instructions are then replaced by those for the selection of the mode of data input and the storage requirements. Key 3 is pressed to show that the data is supplied on cards and no storage is required.

When the data cards have been read the region of solution, with automatic triangulation, is displayed in the working area together with the instructions for fitting the triangles more accurately. At this point key 6 is pressed so that a copy of the display file is drawn onto the digital plotter (Figure 7.14). The elements totally outside the region of solution are now removed. Key 1 is pressed and, following the instructions displayed, the integers 1, 2, 3, 9, 10, 20, 90, 91, 92, 93, 99, 100 and 0 are typed on separate lines on the teletype. The region is now automatically redrawn without these external elements. Those elements partly outside the region must now be moved completely within it by relocating external nodes on the boundary. Key 2 is pressed and instructions for moving one node are displayed. When each external node has been moved onto the boundary, key 6 is pressed to plot the display file onto the digital plotter (Figure 7.15).

We shall not choose to halve any triangles so the next step is to move to the equation definition by pressing key 3. Instructions to type values for 'F' and then 'G' (where 'F' and 'G' are defined at the top of the screen) are now displayed. The values 2.0 then 0.0 are typed on separate lines on the teletype.

Values for G and F are now displayed on the screen with the region of solution divided into triangles and each node numbered. The options to continue or plot are available. Key 1, to continue, is selected. It is now necessary to define the boundary conditions. The instructions described in Chapter 6.5.2 are displayed together with the region, triangles and node numbers. Key 1 is selected for a constant boundary value. The user is now instructed to type this constant on the teletype. The value 0.0 is therefore supplied. Instructions are also displayed to type all the boundary node numbers, ending with a zero. The following numbers are then typed,

TO SOLVE $-\frac{C}{DX}\frac{(DU)}{(DX)} - \frac{D}{DY}\frac{(DU)}{(DY)} + G*U = F$

FIT TRIANGLES TO REGION

KEY 1) REMOVE EXTERNAL ELEMENTS
2) MOVE EXT NODE TO BNDRY (AFTER 1)
3) CONTINUE (TO EQN. DEFINITION)
4) ADD NODE BISECTING AN EDGE
5) MOVE TO NEXT DATA SET
6) PLOT
8) STOP

PROBLEM NAME -

CIRCLER1

NO. OF TRIANGLES

100

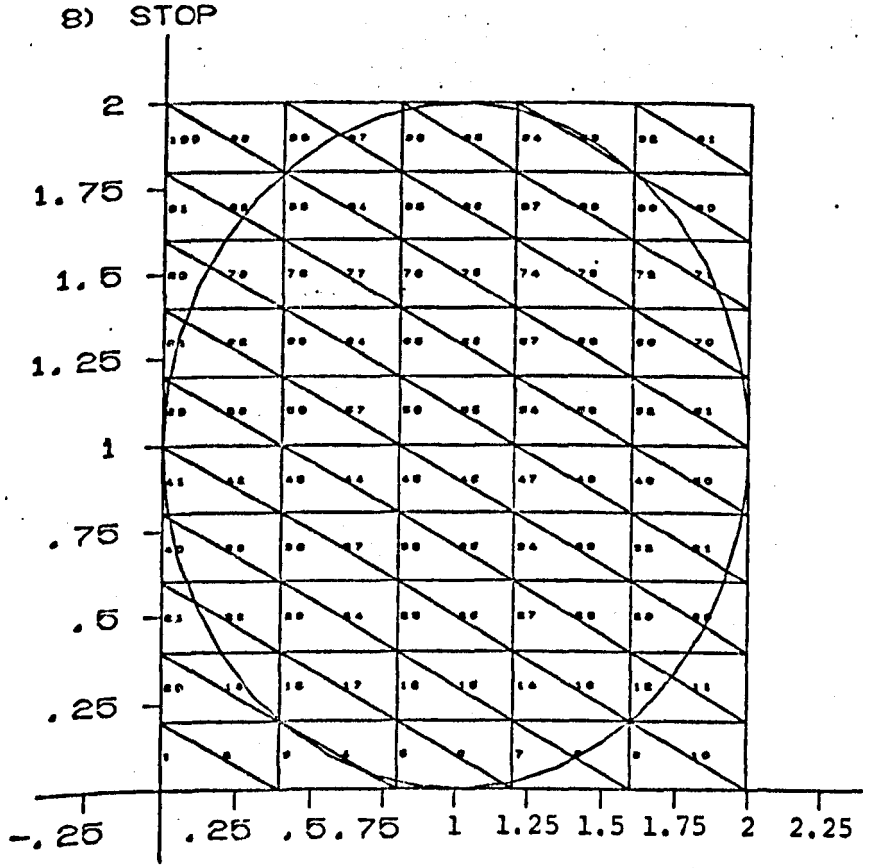


Figure 7.14.

TO SOLVE $-\frac{D}{DX}\frac{(DU)}{(DX)} - \frac{D}{DY}\frac{(DU)}{(DY)} + G*U = F$

FIT TRIANGLES TO REGION

KEY 1) REMOVE EXTERNAL ELEMENTS
2) MOVE EXT NODE TO BNDRY (AFTER 1)
3) CONTINUE (TO EQN. DEFINITION)
4) ADD NODE BISECTING AN EDGE
5) MOVE TO NEXT DATA SET
6) PLOT
8) STOP

PROBLEM NAME -

CIRCLER1

NO. OF TRIANGLES

88

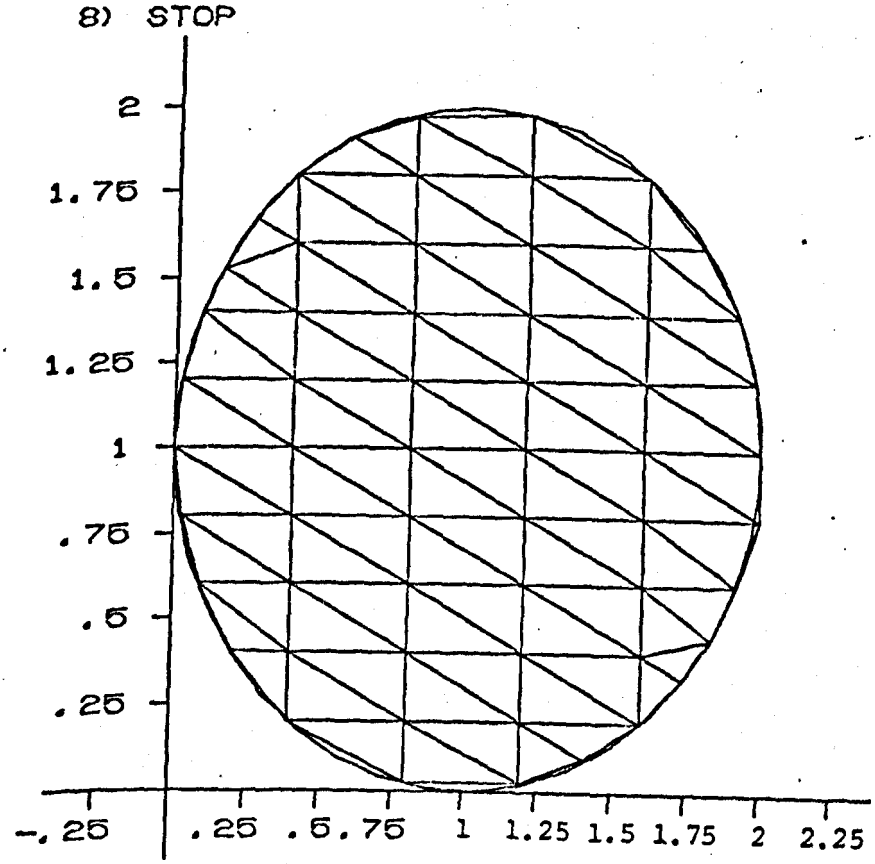


Figure 7.15.

TO SOLVE $-\frac{D}{DX}\frac{(DU)}{(DX)} - \frac{D}{DY}\frac{(DU)}{(DY)} + G*U = F$

END OPTIONS

KEY 1) MOVE TO NEXT DATA SET

2) PLOT

3) ROTATE XY PLANE-TYPE NO. DEGS (IO)
(TYPE O FOR ORIGINAL VIEW)

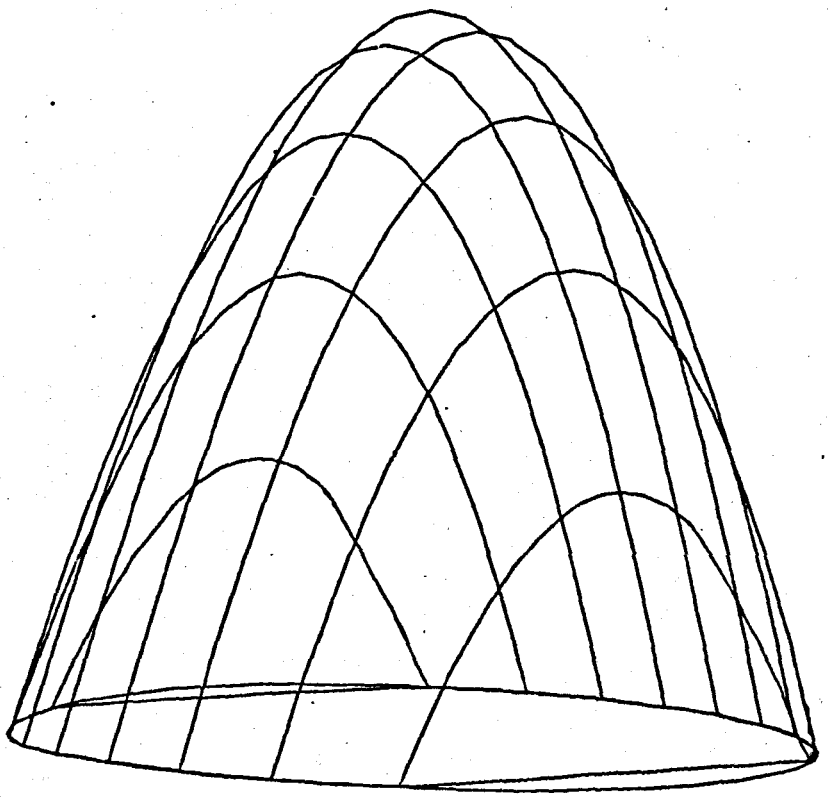
4) RESCALE

5) DISPLAY CONTOUR MAP

8) STOP

PROBLEM NAME -

CIRCLER1



0 <= X <= 1.9966

.00088 <= Y <= 1.9991

0 <= U <= .47908

Figure 7.16.

TO SOLVE $-\frac{D}{DX}\left(\frac{DU}{DX}\right) - \frac{D}{DY}\left(\frac{DU}{DY}\right) + G*U = F$

END OPTIONS

KEY 1) MOVE TO NEXT DATA SET
2) PLOT
6) DISPLAY PROJECTION OF SOLN.
8) STOP

PROBLEM NAME -

CIRCLER1

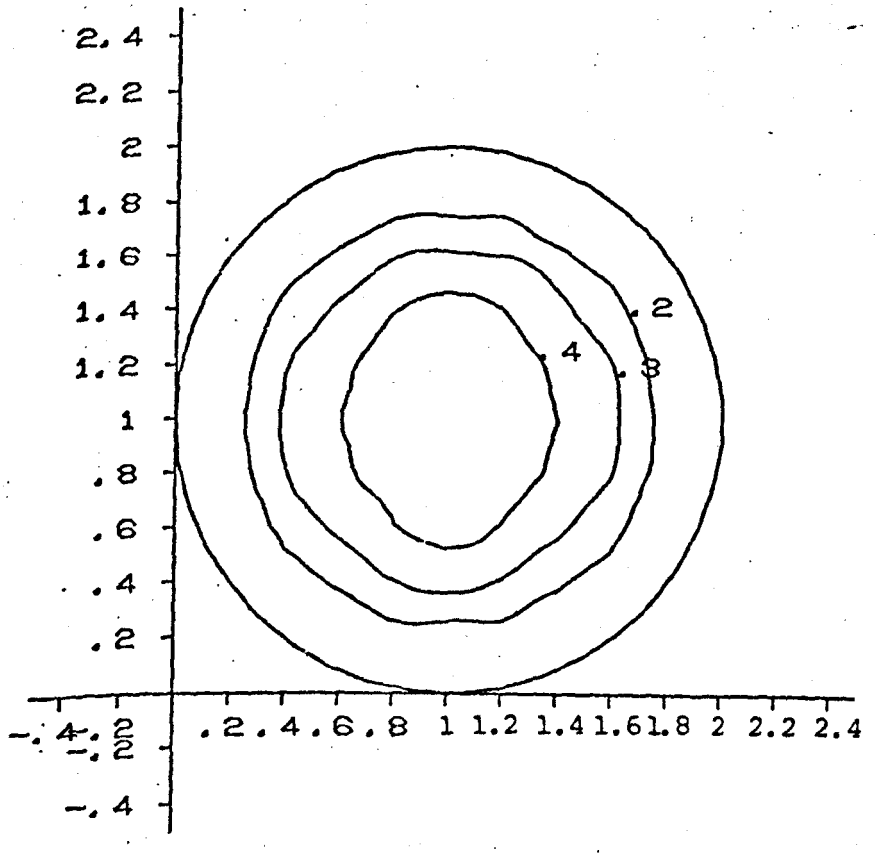


Figure 7.17.

each on a separate line: 1, 2, 4, 6, 7, 9, 10, 14, 15, 20, 22, 26, 27, 32, 34, 38, 39, 44, 46, 50, 51, 52, 55, 56, 57, 58 and 0.

When the boundary conditions are completely defined, control moves to the computation stage and the solution is evaluated automatically. The solution surface is then displayed as a projection onto the plane of the screen and control moves to the finalisation stage and the 'end' options. Key 2 is pressed and the display file, containing the solution surface and the 'end' options, is plotted on the digital plotter (Figure 7.16). At this stage alternative ways of viewing the solution are available. The solution surface can be rotated or a contour map may be drawn. Figure 7.17 shows a contour map with contours drawn at values .2, .3 and .4 of u. This is obtained by pressing key 5 before typing the number and values of the required contours as instructed on the display screen. The program is terminated by pressing key 8.

8. CONCLUSIONS

A closer examination is now given, in the light of the examples in the previous chapter, of the utility or otherwise of incorporating interactive graphical techniques in programs to solve differential equations. Attention is also given to ways in which the approaches taken may be improved.

A principal advantage of interaction when solving differential equations is that the user can be presented, at various stages in a program, with a large number of options, relating to run-time operations, from which selection can be made without the necessity of complex data preparation. In addition these selections can, wherever relevant, be based on intermediate results or information displayed in numerical or graphical form. The speed and compactness with which information can be written or represented on the screen makes this interaction much more efficient than on a conventional terminal where the time taken to output instructions is a major limiting factor. A further advantage is that data can be supplied interactively, either by typing in numbers or expressions on a keyboard, and hence reducing data preparation time, or using a combination of the light-pen and sense keys to 'draw' on the display screen. In this way a complex region shape can often be described more easily, though perhaps less accurately, than using numerical data.

Interaction also simplifies the data validation process. Problem definition and computational parameters are displayed on the screen so that errors can be identified and corrected before computation commences. This eliminates the computer time wasted when incorrect data is supplied for a batch run. Interactive control of graphical output is of considerable use as it is often only possible to decide how best to view solutions after

some visual representation has already been displayed. In the case of ordinary differential equations it is useful to be able to change the variables selected for numerical and graphical display, and for partial differential equations solution surfaces can be rotated or replaced by contour plots.

The major contribution made by purely graphical techniques is in the visual representation of solutions and errors in the form of curves or surfaces. Although this often involves a loss of detail and accuracy it gives a more compact overall picture than is obtained from purely numerical output.

An alternative approach to the 'complete system' for solving differential equations, taken in this thesis is to provide an interactive graphical framework, so that, for example, a user can provide his own 'method of solution' routine, and thus broaden the range of problems for which the system can be used. However, the programming effort involved in setting up this kind of framework for differential equations would be even greater than for the set of programs described here.

The finite element program, although only used for small problems requires the user to wait while the stiffness equation is being solved. This difficulty could be overcome by splitting the program into a pre-processor for setting up the problem, and a post-processor for viewing and manipulating the solution surface.

APPENDIX A. DISPAC

This package consists of routines enabling Fortran programmers to use the graphical display unit more easily than is possible with the basic ICL software. All routines in the package accessible to users have names beginning with the letter G, and there are also some internal routines beginning with Z. Thus users are advised to avoid using names beginning with G or Z.

The graphical display unit is not as straightforward as a digital plotter since the picture must be continually refreshed. Everything to be displayed is held in a display file (an array set up at the start of a program) in the form of items. Each item consists of code which has been generated by routines in the package and placed in a code buffer (another array which is set up for each item). However, items are not displayed until the code in the code buffer is inserted into the display file. This also releases the code buffer for further use. Although each routine in the package could create and insert an item in the display file thus causing the item to appear immediately and make the display unit behave like a digital plotter, this would lose valuable facilities. Chief of these is the ability to build up and identify logically coherent parts of a picture into a single item which can subsequently be manipulated or changed without affecting other parts of the picture. Since this implies that items are not always displayed in the order they are created, the "current position" of the beam is not a useful concept unless the beam is moved to a known position at the start of each item. The routine GNITEM ensures this, and also allows an item to be moved about the screen without recreating it (GNMOVIT).

The display package works entirely in vectors (relative displacements) except for the initial positioning command in GNITEM, so that full use can

be made of the facilities to include subpictures in an item. These subpictures have only one copy which is referred to repeatedly thus saving space with repeated shapes.

The other distinctive features of the display is the ability to interact with the program using either a light pen or sense keys. In either case, a routine (GACT) exists which causes the program (but not the display) to halt until some specified action is taken. This action can then be examined and the display altered appropriately.

The light pen can be used in two ways. Firstly, it may be used to point to a particular item which can then be identified within the program. Alternatively, a tracking cross can be moved around the screen using the light pen, and its co-ordinates determined. Apart from the sense keys on the display unit itself, the sense keys on the operator's console may also be examined, but changing them does not constitute an action for GACT.

Most of the routines are held in the ordinary Fortran library under the user identifier FOR OBJ and are fetched automatically. Those specific to the display device, including GSTART which must always be called, are held on the system disc under the user identifier DISPAC. By assigning channel 3 to one of these before compilation or loading, the appropriate routines will be fetched. For example:

```
&JOB;<user name>;<job name>;
&ASSIGN;3;DC;0;GSTART,DISPAC;
&LOAD;DISPROG,<user name>;DC;91;FORTRAN;
&RUN;
&END;
```

Large programs may have to use the DES1 operating system, in which case the console on which input can be typed for GACT is the operator's console. Smaller programs will run under the DES2 operating system, when

the remote console beside the display unit is used. Since under DES2 the card reader may be in use, the BATCH system has been modified so that jobs can be run from a disc file instead of cards. This is done by typing BATCH,DC. instead of BATCH. after which the logical volume number of the disc and the user identifier and filename of the file must be supplied. Such files contain images of the commands that would have been on cards.

APPENDIX B. NUMERICAL METHODS

In this section the numerical methods of solution used in GIDODE, GIDPDE and FESOLV will be outlined. The methods used in GIDODE are those referred to as Adams',[25], Gear's[25] and the Merson-Newton iteration[30] methods. In GIDPDE the Crank-Nicolson[16] and Peaceman-Rachford[48] finite difference methods are implemented, and in FESOLV a finite element method (54) using triangular elements and linear trial functions is used.

Adams' Method

Consider the ordinary differential equation

$$y' = f(y,t)$$

The Adams-Bashforth p^{th} order predictor equation is

$$y_{n,(0)} = y_{n-1} + \beta_1 h\, y'_{n-1} + \ldots + \beta_p\, h\, y'_{n-p} \qquad (1)$$

where $y_i = y(t_i)$, $t_i = ih$ (h is the step size), $y_i' = f(y_i,t_i)$ and the β_i are given, for example, in "Discrete Variable Methods in Ordinary Differential Equations" by P. Henrici (Wiley, New York, 1962, Chapter 5). The approximation $y_{n,(0)}$ is used in this predictor-corrector scheme as the first approximation in the Adams-Moulton corrector formula of p^{th} order given by

$$y_{n,(m+1)} = y_{n-1} + \beta_0^*\, h\, f(y_{n,(m)},t_n) + \beta_1^*\, h\, y'_{n-1}$$

$$+ \ldots\ldots \qquad \ldots\ldots + \beta^*_{p-1}\, h\, y'_{n-p+1} \qquad (2)$$

The coefficients β_i^* can also be found in the above book by Henrici.

If the corrector equation (2) is iterated until it converges to y_n (as is guaranteed for small enough h and smooth function f) the truncation error introduced in the n^{th} step of the integration will be $C^A_{p+1}\, h^{p+1}\, y^{(p+1)}(t_n) + \theta(h^{p+2})$, where $y^{(k)}$ is the k^{th} derivative of y. The coefficients C^A_{p+1} are given in Henrici where they are called γ_p^*.

Gear's Method

Again consider the equation

$$y' = f(y,t)$$

When this equation is stiff Gear's method can be used. It takes a similar form to Adams' method. The p^{th} order predictor is of the form

$$y_{n,(0)} = \alpha_1 y_{n-1} + \cdots + \alpha_p y_{n-p} + \eta_1 h y'_{n-1}$$

and the corrector

$$y_{n,(m+1)} = \alpha_1^* y_{n-1} + \cdots + \alpha_p^* y_{n-p} + \eta_0^* h f(y_{n,(m)}, t_n)$$

The integration truncation error when the corrector equation is iterated to convergence is

$$C^S_{p+1} h^{p+1} y^{(p+1)}(t_n) + \theta(h^{p+1})$$

where $C^S_{p+1} = 1/(p+1)$.

The α_1^* and η_0^* are given in Henrici.

Merson-Newton Iteration Method

The problem considered is a set of first order ordinary differential equations with conditions specified at two boundary values of the independent variable (a and b) and with the solution required in the range (a, b). That is:

$$\frac{dy_i}{dx} = f_i\,(x, y_1, y_2 \ldots y_N) \quad i = 1, 2 \ldots N \qquad \text{I}$$

with boundary conditions

$$\begin{aligned} y_i\,(a) &= \alpha_i \qquad i = 1, 2 \ldots k \\ y_i\,(b) &= \beta_i \qquad i = k+1, \ldots N \end{aligned} \qquad \text{II}$$

so that there are k boundary conditions at $x = a$ and $N - k$ at $x = b$, where $0 < k < N$.

The first step when solving these equations is to estimate the values of $y_i(a)$ and $y_i(b)$ not specified by II and then integrate the system forwards from $x = a$ and backwards for $x = b$. The problem then remains of improving the initial estimates until the two solutions match. There are N boundary values specified initially and so N unknown values have to be estimated to start the process.

Let us take

$$\begin{aligned} y_j\,(a) &= p_j \qquad j = k+1, \ldots N \\ y_j\,(b) &= p_j \qquad j = 1, 2 \ldots k \end{aligned}$$

The forward integration starts with the given values $\alpha_1, \alpha_2, \ldots \alpha_k$ and the estimates $p_{k+1}, \ldots p_N$ as initial conditions at $x = a$, and produces

a solution $\underline{y}_a(x;\underline{p})$. Similarly the backward integration produces the solution $\underline{y}_b(x;\underline{p})$. The two solutions are computed as far as some matching point $x = m$ in (a, b). The method of integration used is the Runge-Kutta-Merson Method[40].

At the matching point we get the following equations for determining $\underline{p}$

$$\underline{F}(\underline{p}) \equiv \underline{y}_a(m;\underline{p}) - \underline{y}_b(m;\underline{p}) = \underline{0} \qquad \text{III}$$

In general these are non-linear equations in the N variables p_j, which have to be solved iteratively.

If the matching point m is chosen to be at a or b, the number of parameters and equations in III is reduced. However, it is more convenient to discuss the method in general terms and to assume that m is taken at a general point between a and b, giving a full set of N equations at the matching point. The reason for not choosing m at either a or b in all cases is to overcome certain practical problems which may arise in the integration of I. If the system suffers from inherent instability in one direction it can be integrated in the reverse direction with the matching point chosen at the appropriate end. This is not possible when the system is unstable in both directions, but the effect may be mitigated by moving the matching point towards the middle of the range.

Equation III is solved using the modified Newton method:

$$\underline{p}^{(t+1)} = \underline{p}^{(t)} - \lambda_t \left[J(\underline{p}^{(t)})\right]^{-1} \underline{F}(\underline{p}^{(t)})$$

where λ_t is a damping factor (usually ≤ 1) chosen to ensure that $||\underline{F}(\underline{p}^{(t+1)})|| < ||\underline{F}(\underline{p}^{(t)})||$.

$J(\underline{p})$ is the Jacobian matrix $\frac{\partial F_i}{\partial p_j}$ evaluated for parameters p.

The calculations are carried out in the following way.

1. An initial guess $\underline{p}^{(o)}$ is made and t is set to zero.

2. The matrix $J(\underline{p}^{(t)})$ and its triangular factors L_t, U_t are evaluated using Gaussian elimination with partial pivoting so that

$$P_t \, J(\underline{p}^{(t)}) = L_t \, U_t$$

where P_t is a permutation matrix.

3. $\underline{z}$, $\underline{w}$ are evaluated from

$$L_t \, \underline{z} = - P_t \, \underline{F}(\underline{p}^{(t)}), \quad U_t \, \underline{w} = \underline{z}$$

4. The calculations

$$\underline{p}_i = \underline{p}^{(t)} + \lambda_i \underline{w}, \qquad \lambda_t = \frac{1}{2^{i-1}}$$

are carried out for i = 1 , 2 ... until

$$||\underline{F}(\underline{p}_i)||_2 < ||\underline{F}(\underline{p}^{(t)})||_2$$

when $\underline{p}^{(t+1)}$ is set to $\underline{p}_i$ and we proceed to 5.

5. If $||\underline{p}^{(t)} - \underline{p}^{(t+1)}||_\infty < e_1$ and $||\underline{F}(\underline{p}^{(t+1)})||_2 < e_2$ the calculation ends with $\underline{p}^{(t+1)}$ as the final value.

If $||\underline{F}(\underline{p}_1)||_2^2 \leq 0.1 \, ||\underline{F}(\underline{p}^{(t)})||_2^2$ then P_{t+1} is set to P_t, $L_{t+1} = L_t$, $U_{t+1} = U_t$ and t is replaced with t+1 before returning to step 4.

Otherwise we return to step 2 with t replaced by t+1.

Crank-Nicolson Method

Consider the problem

$$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2} + b\frac{\partial u}{\partial x} \qquad \text{for } 0 \leqslant x \leqslant 1 \text{ and } t \geqslant 0$$

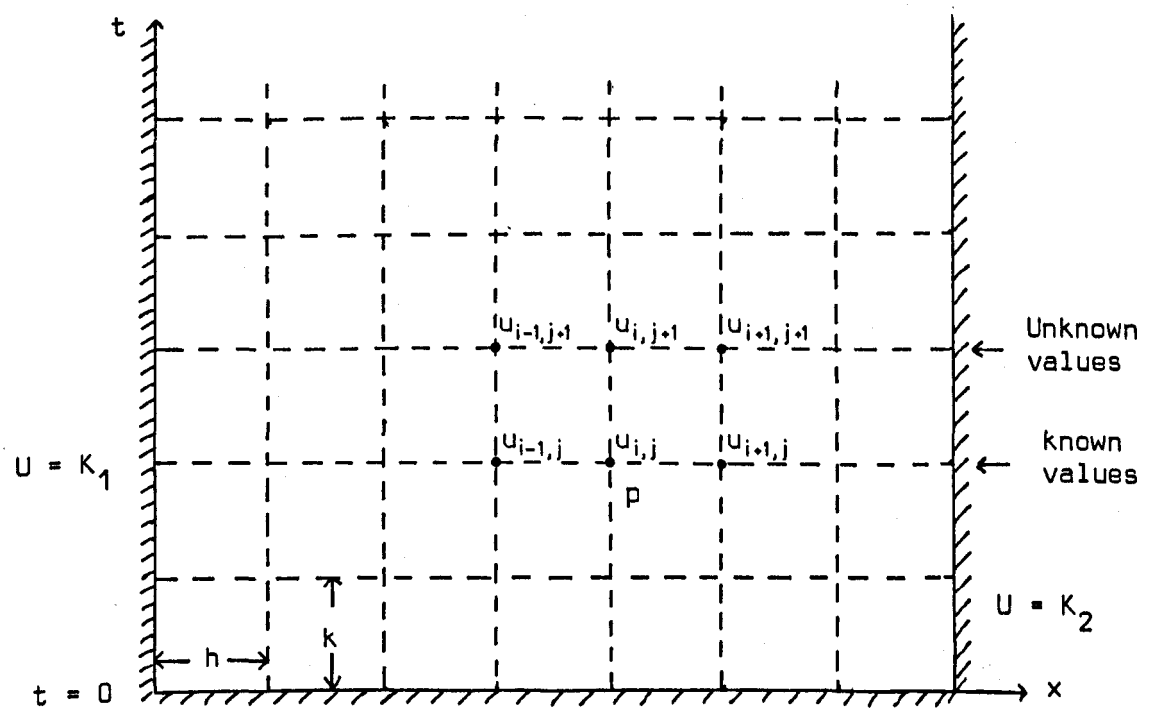
where $u(0,x) = f(x)$

$u(t,0) = K_1$

$u(t,1) = K_2$

and a and b are functions of x and t only.

The region of solution $0 \leqslant x \leqslant 1$, $t \geqslant 0$ can be fitted with a mesh or grid in the following way:



Each rectangle in the mesh has sides $\delta x = h$, $\delta t = k$. The co-ordinates (x,t) of the representative mesh point P are

$$x = ih, \quad t = jk$$

where i and j are integers. The value of u at P is denoted by

$$u_p = u(ih, jk) = u_{i,j}$$

The Crank-Nicolson scheme is implicit and involves expressing the three unknown values $u_{i-1,j+1}$, $u_{i,j+1}$ and $u_{i+1,j+1}$ in terms of the known values $u_{i-1,j}$, $u_{i,j}$ and $u_{i+1,j}$.

Each term in the equation above is replaced in the following way:

$$\frac{\partial u}{\partial t} \simeq \frac{u_{i,j+1} - u_{i,j}}{k}$$

$$a \frac{\partial^2 u}{\partial x^2} \simeq \frac{1}{2} \left\{ \frac{a_{i,j+1}}{h^2} (u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}) + \frac{a_{i,j}}{h^2} (u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) \right\}$$

$$b \frac{\partial u}{\partial x} \simeq \frac{1}{2} \left\{ \frac{b_{i,j+1}}{2h} (u_{i+1,j+1} - u_{i-1,j+1}) + \frac{b_{i,j}}{2h} (u_{i+1,j} - u_{i-1,j}) \right\}$$

which give the system of equations:

$$- k \left(\frac{a_{i,j+1}}{h^2} + \frac{b_{i,j+1}}{2h}\right) u_{i+1,j+1} + \left(\frac{2ka_{i,j+1}}{h^2} + 2\right) u_{i,j+1}$$

$$- k \left(\frac{a_{i,j+1}}{h^2} - \frac{b_{i,j+1}}{2h}\right) u_{i-1,j+1}$$

$$= k \left(\frac{a_{i,j}}{h^2} + \frac{b_{ij}}{2h}\right) u_{i+1,j} - \left(\frac{2ka_{i,j}}{h^2} - 2\right) u_{i,j} + k \left(\frac{a_{ij}}{h^2} - \frac{b_{ij}}{2h}\right) u_{i-1,j} \qquad \text{I}$$

where for each time step there are N-2 equations in the unknown values $u_{i,j+1}$, $i = 2, \ldots N-1$ where $N = \frac{1}{h} + 1$, and where the known boundary values, and values at the previous time step have been incorporated.

Derivative boundary conditions of the form

$$\frac{\partial u}{\partial x} \pm pu = \lambda(t)$$

where λ is bounded and continuous, and p is a known constant, can also be incorporated.

The derivative is replaced by

$$\frac{\partial u_{m,s}}{\partial x} = \frac{1}{2h}\left(u_{m+1,s} - u_{m-1,s}\right) \qquad \text{where } m = 0,N \quad \text{and} \quad s = j,j+1 \qquad \text{II}$$

At grid points on the boundary equations I and II are used to eliminate $u_{-1,s}$ and/or $u_{N+1,s}$ where $s = j,j+1$ and so the total number of equations being solved is increased by the number of boundaries with derivative boundary conditions.

Peaceman-Rachford Method

We now consider the equation

$$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2} + b\frac{\partial^2 u}{\partial y^2}$$

for $0 \leq x \leq 1$, $0 \leq y \leq 1$ and $t \geq 0$.

Where u, a and b are functions of x, y and t only.

The initial conditions are of the form

$$u(x,y,0) = g(x,y) \quad \text{and}$$

boundary conditions

$$u(x,1,t) = K_1$$

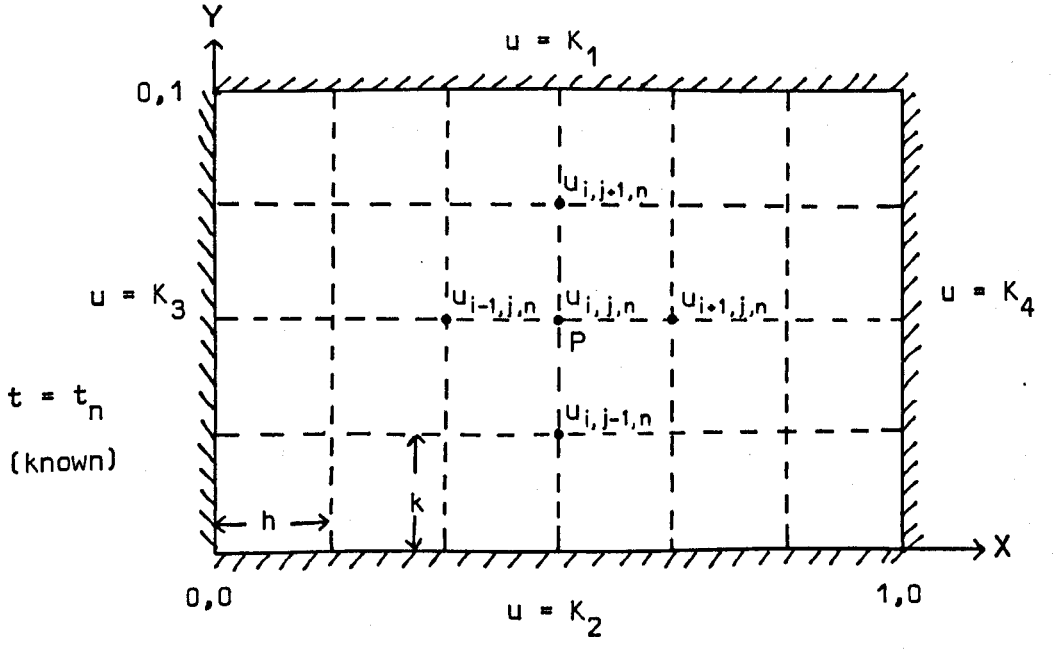
$$u(x,0,t) = K_2$$

$$u(0,y,t) = K_3$$

and

$$u(1,y,t) = K_4$$

The region of solution, $0 \leq x \leq 1$, $0 \leq y \leq 1$ and $t \geq 0$ can be fitted with a mesh similar to the one-dimensional case. For each time step, the grid will be as shown below.



Each cuboid in the mesh will have sides $\delta x=h$, $\delta y=k$, $\delta t=l$. The co-ordinates (x,y,t) of the representative point P are

$$x = ih, \quad y = jk, \quad t = nl$$

where i, j and n are integers. The value of u at P is denoted by $u_p = u_{i,j,n} = u(ih,jk,nl)$.

Assume that the solution is known for time $t = nl$. The Peaceman-Rachford method then consists of replacing only derivatives in one direction, say, along x, by an implicit difference approximation in terms of unknown values of u from the $(n+1)^{th}$ time-level, the derivatives in the other direction, along y, being replaced by an explicit finite-difference approximation. If we have $(N-1)h=1$ and $(M-1)k=1$, that is, N grid points along x and M along y, then applying the corresponding finite difference equation to each of the N-2 internal mesh points along a row parallel to the x-axis gives N-2 equations for the N-2 unknown values of u at these mesh points for time $t = (n+1)l$. The M-2 rows parallel to the x-axis involve the solution of M-2 independent systems of equations, each system containing N-2 unknowns.

The advancement of the solution to the $(n+2)^{th}$ time-level is then achieved by replacing derivatives in the y-direction by an implicit approximation and those along x by an explicit one. The finite difference equation corresponding to each mesh point along columns parallel to the y-axis can then be written giving N-2 independent systems of equations, each involving M-2 unknowns.

The time step, l, must be the same for each advancement, and the solution must be alternated between rows and columns as described above.

The scheme for the equation described above, at the time level $t = (n+1)l$ is as follows:

$$- 1 \left(\frac{a_{i,j,n+1}}{h^2}\right) u_{i+1,j,n+1} + \left(\frac{2l}{h^2} a_{i,j,n+1} + 2\right) u_{i,j,n+1} - 1 \left(\frac{a_{i,j,n+1}}{h^2}\right) u_{i-1,j,n+1}$$

$$= \frac{1}{k^2} b_{i,j,n} \left(u_{i,j+1,n} - 2u_{i,j,n} + u_{i,j-1,n}\right)$$

For the $(n+2)^{th}$ time level the equation at each point is

$$- \frac{1}{k^2} b_{i,j,n+2} \, u_{i,j+1,n+2} + \left(\frac{2l}{k^2} b_{i,j,n+2} + 2\right) u_{i,j,n+2} - \frac{1}{k^2} b_{i,j,n+2} \, u_{i,j-1,n+2}$$

$$= \frac{1}{h^2} a_{i,j,n+1} \left(u_{i+1,j,n+1} - 2u_{i,j,n+1} + u_{i-1,j,n+1}\right)$$

Derivative boundary conditions can be introduced in the same way as for the Crank-Nicolson method.

Finite Element Method

We shall consider the Dirichlet problem, governed inside a domain by Poisson's equation

$$-\Delta u = f \quad \text{in} \quad \Omega$$

and on the boundary by the Dirichlet condition

$$u = 0 \quad \text{on} \quad \Gamma$$

where $-\Delta = -\frac{\partial^2}{\partial x^2} - \frac{\partial^2}{\partial y^2}$

The problem can also be formulated as a variational one where the solution u is the one which minimises the quadratic functional

$$I(v) = \iint_{\Omega} (v_x^2 + v_y^2 - 2fv)\, dx\, dy$$

The region of solution Ω is divided into triangles, and the most simple of trial functions $v^h = a_1 + a_2 x + a_3 y$ is used. ($h = \max h_j$ where h_j is the longest edge of the j^{th} triangle). These functions will be linear inside each triangle and continuous across each edge. Their simplicity lies in the fact that the three coefficients a_1, a_2 and a_3 are uniquely determined by the values of v^h at the three vertices.

If we let $\phi_j(x,y)$ be the trial function which equals 1 at the j^{th} node and zero at all other nodes, then pyramid functions ϕ_j form a basis for the trial space S^h.

An arbitrary v^h in S^h can be expressed as the linear combination

$$v^h(x,y) = \sum_{j=1}^{N} q_j \phi_j(x,y)$$

where the co-ordinate q_j has direct physical significance as the displacement v^h at the j^{th} node $z_j = (x_j, y_j)$.

The optimal co-ordinates Q_j are now determined by minimising

$$I(v^h) = I(\Sigma q_j \phi_j)$$

which is quadratic in $q_1, q_2 \ldots q_N$.

We have

$$I(v) = \iint_\Omega ((v_x^2 + v_y^2) - 2fv)\, dx\, dy$$

$$= a(v,v) - 2(f,v)$$

Substituting the trial functions v into the strain energy a(v,v) yields a quadratic in the co-ordinates q_j

$$a(v,v) = q^T K q = \sum_e q_e^T K_e q_e$$

where the entries of K are the inner products $K_{jk} = a(\phi_K, \phi_j)$, and $\sum_e$ is the sum over all elements (i.e. triangles).

To evaluate K_e it is necessary to find the nodal parameter $q_e^{(3)}$ in terms of the coefficients a_i.

We have

$$q = GA$$

i.e.
$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} 1 & X_1 & Y_1 \\ 1 & X_2 & Y_2 \\ 1 & X_3 & Y_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Inverting G

$$A = Hq_e$$

where

$$H = \frac{1}{\det(G)} \begin{bmatrix} G_{11} & G_{21} & G_{31} \\ G_{12} & G_{22} & G_{32} \\ G_{13} & G_{23} & G_{33} \end{bmatrix} \qquad \text{(I)}$$

$$G_{11} = X_2 Y_3 - Y_2 X_3 \qquad \text{etc.}$$

We now calculate the energy integral in terms of the coefficients a_i

$$a_e(v,v) = \iint_e (v_x^2 + v_y^2)\, dX\, dY$$

$$= A^T N A \qquad \text{(II)}$$

$$= (a_2^2 + a_3^2) \iint_e dX\, dY$$

$$= A^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & \Delta e & 0 \\ 0 & 0 & \Delta e \end{bmatrix} A$$

Therefore

$$N = \Delta e \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

From I and II

$$a_e(v,v) = q_e^T H^T N H q_e$$

and

$$K_e = H^T N H$$

$$= \frac{\Delta_e}{\det^2(G)} \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} G_{11} & G_{21} & G_{31} \\ G_{12} & G_{22} & G_{32} \\ G_{13} & G_{23} & G_{33} \end{bmatrix}$$

To evaluate the load vector F we have

$$\iint_{\Omega} fv = q^T F = \sum_e q_e^T F_e = \sum_e \iint_e fv$$

$$\iint_e fv = a_1 \iint f + a_2 \iint fX + a_3 \iint fY$$

$$= A^T \sigma$$

$$= q_e^T H^T \sigma$$

where $\sigma_1 = f\Delta e$, $\sigma_2 = 0$, $\sigma_3 = 0$

$$F_e = H^T \sigma = \frac{\Delta e}{\det(G)} \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$= \frac{\Delta e}{\det(G)} \begin{bmatrix} G_{11} \\ G_{21} \\ G_{31} \end{bmatrix}$$

The optimal co-ordinates Q_j are now determined by solving the linear algebraic equations

$$q^T K q = q^T F$$

That is: $$KQ = F$$

REFERENCES

1. Ahlberg, J. H., Nilson, E. N., Walsh, J. L. 'The Theory of Splines and their Applications'. Academic Press, New York, 1967.

2. Atkinson, R. C., Wilson, H. A. 'Computer-Assisted Instruction - A Book of Readings'. Academic Press, New York, 1969.

3. Baker, C. L. 'JOSS - Introduction to a Helpful Assistant'. Memorandum RM-5058-PR, RAND Corporation, Santa Monica, California, July 1966.

4. Barron, D. W., Buxton, J. N., Hartley, D. F., Nixon, E., Stachey, C. 'The Main Features of C.P.L.' Comp. Jour., 6, 2, 1963.

5. Barry, C. D., Ellis, R. A., Graesser, S. M., Marshall, G. R. 'Display and Manipulation in Three Dimensions' in 'Pertinent Concepts in Computer Graphics'. Ed. Faiman, M., Nievergelt, J. Univ. of Illinois Press, Urbana, 1969.

6. Berk, T. (Ed.) Proceedings ACM Symposium on Graphics Languages, April 1976, Miami Beach, Florida.

7. Bitzer, D. L., Slottow, H. G. 'The Plasma Panal - A Digital Addressable Display with Inherent Memory'. FJCC 1966.

8. Bjurel, G., Dahlquist, G., Lindberg, B., Linde , S. 'Survey of Stiff Ordinary Differential Equations'. Report NA 70.11, Dept. of Information Processing, The Royal Institute of Technology, Stockholm, Sweden, 1970.

9. Burtnyk, N., Wein, M. 'Computer Animation of Free Form Images'. Computer Graphics 9, 1, Spring 1975.

10. Chasen, S. H. 'Applications of Man-Computer Graphics' in 'Computer Graphics in Management'. Ed. Parslow, R. D., Green, R. Plenum Press, 1971.

11. Chau, A. Y. C., Davies, B. W., Zacharov, B. 'Island - An Interactive Graphics System for Mathematical Analysis'. Comp. Jour. 17, 2, 1974.

12. Collins, D. W. 'A Computer Graphics System for Modular Building Elevation Design'. Computer Graphics, 9, 1, Spring 1975.

13. Cottafava, G., Le Moli, G. 'Automatic Contour Map'. Comm. ACM 12, 7, July 1969.

14. Craigie, J. A. I. 'A Variable Order Multistep Method for the Numerical Solution of Stiff Systems of Ordinary Differential Equations'. Numerical Analysis Report No. 11, Dept. of Mathematics, Univ. of Manchester, Aug. 1975.

15. Crane, C. M. 'Contour Plotting for Functions specified at Nodal Points of an Irregular Mesh based on an Arbitary Two Parameter Co-ordinate System'. Alg. 75, Comp. Jour. 15, 4, 1972.

16. Crank, J., Nicolson, P. 'A Practical Method for Numerical Evaluation of Solutions of Partial Differential Equations of the Heat Conduction Type'. Proc. Camb. Phil. Soc. 43, pp. 50 - 67, 1947.

17. Davis, M. R., Ellis, T.O. 'The Rand Tablet: A Man-Machine Graphical Communication Device'. FJCC 1964.

18. 'DISPAC' - A Graphical Display Package. Computer Centre, Univ. of Keele, Oct. 1973.

19. Elliot Green, R., Parslow, R. D. 'Computer Graphics in Management'. Gower Press, London, 1970.

20. Englebart, D. C., English, W. K. 'A Research Center for Augmenting Human Intellect'. FJCC 1968.

21. English, W. K., Englebart, D. C., Berman, M. L. 'Display-Selection Techniques for Text Manipulation'. IEEE Trans. on Human Factors, HFE-8, 1, 5, 1967.

22. Everett, R. R. 'The Whirlwind 1 Computer'. Review of Electronic Digital Computers, Joint AIEE-IRE Conference 70, Feb. 1952.

23. Faiman, M., Nievergelt, J. 'Pertinent Concepts in Computer Graphics'. Univ. of Illinois Press, 1969.

24. 'FRED' - Fortran IV Routines for the Elliott Display, 4100 Technical Manual, Vol. 2, Part 16, Section 3, ICL 1969.

25. Gear, C. W. 'Numerical Initial Value Problems in Ordinary Differential Equations'. Prentice-Hall, Englewood Cliffs, N.J., 1971.

26. Gear, C. W. Algorithm 407. Comm. of ACM, 14, 3, March, 1971.

27. Giloi, W. K. 'On High-Level Programming Systems for Structured Display Programming'. Computer Graphics, 9, 1, Spring 1975.

28. Gourlay, A. R. 'Hopscotch: A Fast Second-Order P.D.E. Solver'. J. Inst. Maths. Applics., 6, 4, 1970.

29. Hambin, C. L. 'Translation to and from Polish Notation'. Comp. Jour. 5, 3, 1962.

30. Hazelgrove, G. B. 'The Solution of Non-Linear Equations and of Differential Equations with Two-Point Boundary Conditions'. Comp. Jour. 4, 3, 1961.

31. Henrici, P. 'Discrete Variable Methods in Ordinary Differential Equations'. Wiley, New York, 1962.

32. Hooper, R., Toye, I. 'Computer Assisted Learning in the United Kingdom'. Council for Educational Technology, 1975.

33. Hull, T. E. et. al. 'Comparing Numerical Methods for Ordinary Differential Equations'. SIAM J. Numer. Anal., 9, 4, Dec. 1972.

34. Iverson, K. 'A Programming Language'. Wiley, New York, 1962.

35. Jacobs, J. F. 'Practical Evaluation of Command and Control Systems'. Report MTP-7, Bedford, Mass. MITRE Corporation, 1975.

36. Jennings, A. 'Accelerating the Convergence of Matrix Iterative Processes'. J. Inst. Maths. Applics., 8, 1, 1971.

37. Kubert, B., Szabo, J., Giulieri, S. 'The Perspective Representation of Functions of Two Variables'. J. Assoc. Comp. Mach., 15, 2, April, 1968.

38. Lavick, J. J. 'Computer Aided Design at McDonnell-Douglas' in 'Advanced Computer Graphics'. Ed. Parslow, R. D., Elliot Green, R. Plenum Press, 1971.

39. Leninthal, C., Barry, C. D. Ward, S. A., Zwick, M. 'Computer Graphics in Macromolecular Chemistry' in 'Emerging Concepts in Computer Graphics'. Ed. Nievergelt, J., Secrest, D. W. A. Benjamin Inc., New York, 1968.

40. Mayers, D. F. 'Methods of Runge-Kutta Type' in 'Numerical Solution of Ordinary and Partial Differential Equations'. Ed. Fox, L. Pergamon, New York, 1962.

41. McLain, D. H. 'Drawing Contours from Arbitary Data Points'. Comp. Jour. 17, 4, 1974.

42. Mitchell, A. R. 'The Finite Element Method'. I.M.A. 10, 3, March 1974.

43. Morse, S. P. 'Concepts of Use in Contour Map Processing'. Comm. ACM, 12, 3, March 1969.

44. Numerical Algorithms Group Library. The Numerical Algorithms Group Ltd. NAG Central Office, 12 Banbury Road, Oxford.

45. O'Brian, C. D., Brown, H. G. 'Image - A Language for the Interactive Manipulation of a Graphics Environment'. Computer Graphics, 9, 1, Spring 1975.

46. Parslow, R. D., Elliot Green, R. 'Advanced Computer Graphics'. Plenum Press, London and New York, 1971.

47. Parslow, R. D., Prowse, R. W., Elliot Green, R. 'Computer Graphics'. Plenum Press, London and New York, 1969.

48. Peaceman, D. W., Rachford, H. H. 'The Numerical Solution of Parabolic and Elliptic Differential Equations'. J. Soc. Indust. Appl. Maths., 3, 1, 1955.

49. Roberts, L. G. 'Machine Perception of Three Dimensional Solids'. M.I.T. Lincoln Laboratory, TR 315, May 1963.

50. Robertson, H. H. 'Numerical Integration of Systems of Stiff Ordinary Differential Equations with Special Structure'. Num. Anal. Report No. 8, Dept. of Mathematics, Univ. of Manchester, March 1975.

51. Sharpe, W. F., Jacob, N. L. 'An Introduction to Computer Programming in the BASIC Language'. Free Press, New York, 1970.

52. Smith, G. D. 'Numerical Solution of Partial Differential Equations'. Oxford University Press, London, 1965.

53. Simon, R. L. 'Computer Aided Design of Sculptured Surfaces'. Computer Graphics, 9, 1, Spring 1975.

54. Strang, G., Fix, G. J. 'An Analysis of the Finite Element Method'. Prentice-Hall Inc., Englewood Cliffs, N.J., 1973.

55. Sutherland, I. E. 'Sketchpad - A Man-Machine Graphical Communication System'. Proc. SJCC, 1963.

56. Sutherland, I. E. 'A Head-Mounted Three-Dimensional Display'. Proc. FJCC, 1968.

57. Sutherland, I. E., Sproul, R. F., Schumacker, R. A. 'A Characterisation of Ten Hidden Surface Algorithms'. ACM Computing Surveys, 6, 1, March 1974.

58. Warnock, J. E. 'A Hidden-line Algorithm for Computer Generated Half-Tone Pictures'. TR4-15, Computer Science Dept., Univ. of Utah, 1969.

59. Weisberg, D. E. 'Man-Machine Communication and Process Control'. Data Processing Magazine, Sept. 1967.

60. Weissman, C. 'LISP 1.5 Primer'. Dickenson, Belmont, Calif., 1967.

61. Whiteman, J. R. 'The Mathematics of Finite Elements and Applications'. Academic Press, London, 1973.

62. Williams, R. 'A Survey of Data Structures for Computer Graphics Systems'. Computer Surveys, 3, 1, March, 1971.

63. Williamson, H. 'Hidden-Line Plotting Program'. Algorithm 420, Comm. ACM, 15, 2, Feb. 1972.

64. Young, D. M., Gregory, R. T. 'A Survey of Numerical Mathematics'. Volume II, Addison-Wesley Publishing Company, Reading, Mass., 1973.

65. Zienkiewicz, O. C. 'The Finite Element Method in Engineering Science'. McGraw-Hill, London, 1971.

66. Zlámel, M. 'Recent Advances in Finite Elements' in 'The Mathematics of Finite Elements and Applications'. Ed. Whiteman, J. R., Academic Press, London, 1973.